SYBASE®

Enhanced Full-Text Search Specialty Data Store
ユーザーズ・ガイド

Adaptive Server® Enterprise

15.0.2

ドキュメント ID：DC33112-01-1502-01
改訂：2009 年 2 月

マニュアルの注文

マニュアルの注文を承ります。ご希望の方は、サイベース株式会社営業部または代理店までご連絡ください。マニュアルの変更は、弊社の定期的なソフトウェア・リリース時にのみ提供されます。

Sybase, Inc., One Sybase Drive, Dublin, CA 94568.

目次

はじめに

このユーザーズ・ガイドでは、Sybase® Adaptive Server® Enterprise で Enhanced Full-Text Search Specialty Data Store 製品を使用する方法について説明します。このマニュアルには、この製品の拡張版の機能について記載されています。

対象読者

このマニュアルが対象としている読者は、Adaptive Server で Enhanced Full-Text Search Specialty Data Store が使用されるように設定するシステム管理者と、Adaptive Server データの全文検索を実行するユーザです。

このマニュアルの内容

このマニュアルの内容は、次のとおりです。

- 「第 1 章 概要」では、Enhanced Full-Text Search Specialty Data Store の概要について説明します。
- 「第 2 章 拡張型全文検索エンジンについて」では、Enhanced Full-Text Search Specialty Data Store のコンポーネントとその機能について説明します。
- 「第 3 章 Adaptive Server の全文検索の設定」では、Enhanced Full-Text Search Specialty Data Store によってデータベース上で全文検索ができるように Adaptive Server を設定する方法について説明します。
- 「第 4 章 Verity 機能の設定」では、全文検索クエリを発行するために行う必要がある設定について説明します。
- 「第 5 章 全文検索クエリの作成」では、全文検索クエリを記述するために使用するコンポーネントについて説明します。
- 「第 6 章 システム管理」では、システム管理について説明します。
- 「第 7 章 パフォーマンスとチューニング」では、パフォーマンスとチューニングの問題について説明します。
- 「第 8 章 Verity のトピック」では、Verity エンジンの設定について説明します。
- 「付録 A システム・プロシージャ」では、Enhanced Full-Text Search Specialty Data Store のシステム・プロシージャについて説明します。
- 「付録 B サンプル・ファイル」では、*textsvr.cfg* ファイルの例を示し、Enhanced Full-Text Search Specialty Data Store に付随するサンプル・ファイルについて説明します。さらに sample_text_main.sql スクリプトに関しても説明します。

- 「付録 C Unicode のサポート」では、Enhanced Full Text Search Specialty Data Store で Unicode を使用できるように設定する方法について説明します。
- 「付録 D XML データの使用」では、フィールドとゾーンを使用してテキスト・インデックスにインデックス付けした XML データの正しい形式と、そのようなデータを使用する場合のサンプルについて説明します。

関連マニュアル

Sybase® Adaptive Server® Enterprise には次のマニュアルが用意されています。必要に応じて参照してください。

- 使用しているプラットフォームの『リリース・ノート』－ マニュアルには記載できなかった最新の情報が記載されています。

  『リリース・ノート』の最新版 (英語版) にはインターネットからアクセスできます。この製品の CD-ROM がリリースされたあとに追加された重要な製品情報やマニュアル情報を確認する場合は、Sybase Technical Library を参照してください。
- 使用しているプラットフォームの『インストール・ガイド』－ すべての Adaptive Server および関連する Sybase 製品のインストール、アップグレード、設定の手順について説明しています。
- Adaptive Server Enterprise 新機能ガイド－ Adaptive Server バージョン 15.0 の新しい機能について説明しています。また、新しい機能をサポートするためのシステム変更や、既存のアプリケーションに影響する変更についても説明しています。
- 『ASE Replicator ユーザーズ・ガイド』－ プライマリ・サーバから 1 つ以上のリモート Adaptive Server に対して基本的な複写を行うための Adaptive Server の Adaptive Server Replicator 機能の使用方法について説明しています。
- 『コンポーネント統合サービス・ユーザーズ・ガイド』－ リモートの Sybase データベースおよび Sybase 以外のデータベースへ接続するための Adaptive Server コンポーネント統合サービス機能について説明しています。
- 使用しているプラットフォームの『Adaptive Server Enterprise 設定ガイド』－ Adaptive Server の特定の設定作業を行う方法について説明しています。
- 『Full-Text Search Specialty Data Store ユーザーズ・ガイド』－ Verity で全文検索機能を使用して Adaptive Server Enterprise のデータを検索する方法について説明しています。
- 『用語解説』－ Adaptive Server マニュアルで使用されている技術用語について説明しています。
- 『Historical Server ユーザーズ・ガイド』－ Historical Server を使用して、SQL Server® と Adaptive Server のパフォーマンス情報を取得する方法について説明しています。
- 『Adaptive Server Enterprise における Java』－ Adaptive Server データベースで Java クラスをデータ型、関数、ストアド・プロシージャとしてインストールして使用する方法について説明しています。

- 『Job Scheduler ユーザーズ・ガイド』－ コマンド・ラインまたはグラフィカル・ユーザ・インタフェース (GUI) を使用して、ローカルまたはリモートの Adaptive Server でジョブをインストールして設定する方法、および作成してスケジュールする方法について説明しています。
- 『Monitor Client Library プログラマーズ・ガイド』－ Adaptive Server のパフォーマンス・データにアクセスする Monitor Client Library アプリケーションの記述方法について説明しています。
- 『Monitor Server ユーザーズ・ガイド』－ Monitor Server を使用して、SQL Server と Adaptive Server のパフォーマンス統計を取得する方法について説明しています。
- 『パフォーマンス&チューニング・シリーズ』－ Adaptive Server で最高のパフォーマンスを実現するためのチューニング方法について説明しています。このマニュアルは以下の 4 冊に分かれています。
  - 『基本』－ Adaptive Server のパフォーマンスに関する問題の理解と調査の基本について説明しています。
  - 『ロックと同時実行制御』－ さまざまなロック・スキームを使用して Adaptive Server のパフォーマンスを向上させる方法について説明しています。
  - 『クエリ処理と抽象プラン』－ オプティマイザがクエリを処理する方法と抽象プランを使用してオプティマイザのプランの一部を変更する方法について説明しています。
  - 『モニタリング・テーブル』－ パフォーマンスのモニタリングと最適化のために、統計を取得し、使用する方法について説明しています。
- 『クイック・リファレンス・ガイド』－ コマンド、関数、システム・プロシージャ、拡張システム・プロシージャ、データ型、ユーティリティの名前と構文の包括的な一覧表を記載したポケット版のマニュアルです。
- 『ASE リファレンス・マニュアル』－ 詳細な Transact-SQL® 情報を記載しています。このマニュアルは以下の 4 冊に分かれています。
  - 『ビルディング・ブロック』－ Transact-SQL のデータ型、関数、グローバル変数、式、識別子とワイルドカード、予約語。
  - 『コマンド』－ Transact-SQL のコマンド。
  - 『プロシージャ』－ Transact-SQL のシステム・プロシージャ、カタログ・ストアド・プロシージャ、システム拡張ストアド・プロシージャ、dbcc ストアド・プロシージャ。
  - 『テーブル』－ Transact-SQL のシステム・テーブルと dbcc テーブル。

- 『システム管理ガイド』－ サーバとデータベースを管理するための高度な情報について説明しています。このマニュアルでは、物理的なリソース、セキュリティ、ユーザ・データベース、システム・データベースの管理方法、および文字セットの変換、言語の国際化、ソート順の指定方法についての手順とガイドラインを説明しています。
- 『システム・テーブル・ダイヤグラム』－ システム・テーブルと、そのエンティティとの関係をポスター形式で図解しています。印刷版のみが用意されています。
- 『Transact-SQL ユーザーズ・ガイド』－ リレーショナル・データベース言語の Sybase の拡張版である Transact-SQL について説明しています。このマニュアルでは、データベース管理システムの操作に慣れていない方のために、テキストブック形式で説明しています。また、pubs2 と pubs3 サンプル・データベースについても説明します。
- 『Adaptive Server 分散トランザクション管理機能の使用』－ 分散トランザクション処理環境での Adaptive Server DTM 機能の設定、使用、トラブルシューティングについて説明しています。
- 『高可用性システムにおける Sybase フェールオーバの使用』－ Sybase のフェールオーバ機能を使用して、Adaptive Server を高可用性システムのコンパニオン・サーバとして設定する方法について説明しています。
- 『Unified Agent および Agent Management Console』－ Unified Agent について説明します。Unified Agent は、分散 Sybase リソースを管理、モニタ、制御するためのランタイム・サービスを提供します。
- 『ASE ユーティリティ・ガイド』－ オペレーティング・システム・レベルで実行される isql および bcp などの、Adaptive Server のユーティリティ・プログラムについて説明しています。
- 『Web Services ユーザーズ・ガイド』－ Adaptive Server 用の Web サービスの設定、使用、トラブルシューティング方法について説明しています。
- 『XA インタフェース統合ガイド for CICS、Encina、TUXEDO』－ X/Open XA トランザクション・マネージャを備えた Sybase DTM XA インタフェースを使用する方法について説明しています。
- 『Adaptive Server Enterprise における XML Services 』では、データベースに XML 機能を導入する、Sybase ネイティブの XML プロセッサと Sybase Java ベースの XML のサポートについて、また XML サービスに準拠したクエリとマッピング用の関数について説明しています。

その他の情報

Sybase Getting Started CD、SyBooks CD、Sybase Product Manuals Web サイトを利用すると、製品について詳しく知ることができます。

- Getting Started CD には、PDF 形式のリリース・ノートとインストール・ガイド、SyBooks CD に含まれていないその他のマニュアルや更新情報が収録されています。この CD は製品のソフトウェアに同梱されています。Getting Started CD に収録されているマニュアルを参照または印刷するには、Adobe Acrobat Reader が必要です (CD 内のリンクを使用して Adobe の Web サイトから無料でダウンロードできます)。
- SyBooks CD には製品マニュアルが収録されています。この CD は製品のソフトウェアに同梱されています。Eclipse ベースの SyBooks ブラウザを使用すれば、使いやすい HTML 形式のマニュアルにアクセスできます。

  一部のマニュアルは PDF 形式で提供されています。これらのマニュアルは SyBooks CD の PDF ディレクトリに収録されています。PDF ファイルを開いたり印刷したりするには、Adobe Acrobat Reader が必要です。

  SyBooks をインストールして起動するまでの手順については、Getting Started CD の『SyBooks インストール・ガイド』、または SyBooks CD の *README.txt* ファイルを参照してください。
- Sybase Product Manuals Web サイトは、SyBooks CD のオンライン版であり、標準の Web ブラウザを使ってアクセスできます。また、製品マニュアルのほか、EBFs/Maintenance、Technical Documents、Case Management、Solved Cases、Newsgroups、Sybase Developer Network へのリンクもあります。

  Technical Library Product Manuals Web サイトにアクセスするには、Product Manuals (http://www.sybase.com/support/manuals/) にアクセスしてください。

Web 上の Sybase 製品の動作確認情報

Sybase Web サイトの技術的な資料は頻繁に更新されます。

❖ **製品認定の最新情報にアクセスする**

1. Web ブラウザで Technical Documents を指定します。(http://www.sybase.com/support/techdocs/)
2. 左側のナビゲーション・バーから [Products] を選択します。
3. 製品リストから製品名を選択し、[Go] をクリックします。
4. [Certification Report] フィルタを選択し、時間枠を指定して [Go] をクリックします。
5. [Certification Report] のタイトルをクリックして、レポートを表示します。

❖ **コンポーネント認定の最新情報にアクセスする**

1 Web ブラウザで Availability and Certification Reports を指定します。(http://certification.sybase.com/)

2 [Search By Base Product] で製品ファミリとベース製品を選択するか、[Search by Platform] でプラットフォームとベース製品を選択します。

3 [Search] をクリックして、入手状況と認定レポートを表示します。

❖ **Sybase Web サイト (サポート・ページを含む) の自分専用のビューを作成する**

MySybase プロファイルを設定します。MySybase は無料サービスです。このサービスを使用すると、Sybase Web ページの表示方法を自分専用にカスタマイズできます。

1 Web ブラウザで Technical Documents を指定します。(http://www.sybase.com/support/techdocs/)

2 [MySybase] をクリックし、MySybase プロファイルを作成します。

Sybase EBF とソフトウェア・メンテナンス

❖ **EBF とソフトウェア・メンテナンスの最新情報にアクセスする**

1 Web ブラウザで Sybase Support Page を指定します。(http://www.sybase.com/support)

2 [EBFs/Maintenance] を選択します。MySybase のユーザ名とパスワードを入力します。

3 製品を選択します。

4 時間枠を指定して [Go] をクリックします。EBF/Maintenance リリースの一覧が表示されます。

鍵のアイコンは、「Technical Support Contact」として登録されていないため、一部の EBF/Maintenance リリースをダウンロードする権限がないことを示しています。未登録でも、Sybase 担当者またはサポート・コンタクトから有効な情報を得ている場合は、[Edit Roles] をクリックして、「Technical Support Contact」の役割を MySybase プロファイルに追加します。

5 EBF/Maintenance レポートを表示するには [Info] アイコンをクリックします。ソフトウェアをダウンロードするには製品の説明をクリックします。

表記規則

ディレクトリ・パス 読みやすくするために、このマニュアルではディレクトリ・パスは UNIX フォーマットで記述されています。Windows NT の場合は、*$SYBASE* を *%SYBASE%* に、スラッシュ (/) を円記号 (¥) に置き換えてください。たとえば、次のようなユーザ入力があるとします。

```
$SYBASE/$SYBASE_FTS/scripts
```

次のように置き換えます。

```
%SYBASE%¥%SYBASE_FTS%¥scripts
```

SQL 文のフォーマット　SQL は自由な形式の言語で、1 行内のワード数や改行の仕方に規則はありません。このマニュアルでは、読みやすくするため、例や構文を文の句ごとに改行しています。複数の部分からなり、2 行以上にわたる場合は、字下げしています。

SQL の構文規則　構文の規則は、次の表のとおりです。

表 *1:* 構文の表記規則

キー	定義
コマンド	コマンド名、コマンドのオプション名、ユーティリティ名、ユーティリティのフラグ、キーワードは、構文中では次のように太字の Courier で表記し、文中では**太字の** Helvetica で表記する。
変数	変数 (ユーザが入力する値を表す語) は斜体で表記する。
{ }	中カッコは、その中から必ず 1 つ以上のオプションを選択しなければならないことを意味する。コマンドには中カッコは入力しない。
[ ]	角カッコは、オプションを選択しても省略してもよいことを意味する。コマンドには角カッコは入力しない。
( )	このカッコはコマンドの一部として入力する。
|	縦線は、複数のオプションのうち 1 つだけを選択できることを意味する。
,	カンマは、中カッコまたは角カッコの中のカンマで区切られたオプションをいくつでも選択できることを意味する。カンマはコマンドの一部として入力する。

- オプション句のあるコマンドの構文例を示します。

  sp_dropdevice [*device_name*]

  次は、複数のオプション句を持つコマンドの構文の例です。

  select *column_name*
  from *table_name*
  where *search_conditions*

  構文では、キーワード (コマンド) は通常のフォントで表記し、識別子は小文字で表記します。また、ユーザが提供するワードは斜体で表記します。

- Transact-SQL コマンドの使用例は次のように表記します。

```
select * from publishers
```

- コンピュータからの出力例を示します。

```
pub_id      pub_name                  city        state
------      --------------------      ----------  -----
0736        New Age Books             Boston      MA
0877        Binnet & Hardley          Washington  DC
1389        Algodata Infosystems      Berkeley    CA

(3 rows affected)
```

大文字と小文字の区別　このマニュアルでは、例に使用する文字はほとんどが小文字ですが、Transact-SQL のキーワードを入力するときは、大文字と小文字は区別されません。たとえば、SELECT、Select、select はすべて同じです。

テーブル名などのデータベース・オブジェクトの大文字と小文字を Adaptive Server が区別するかどうかは、Adaptive Server にインストールされたソート順によって決まります。シングル・バイト文字セットを使用している場合は、Adaptive Server のソート順を再設定することによって、大文字と小文字の区別の取り扱い方を変更できます。

必須選択肢

- 中カッコと縦線：中からオプションを 1 つだけ選択します。

```
{die_on_your_feet | live_on_your_knees | live_on_your_feet}
```

- 中カッコとカンマ：1 つまたは複数のオプションを選択します。複数のオプションを選択する場合は、それぞれをカンマで区切ります。

```
{cash, check, credit}
```

任意選択肢

- 角カッコ内の *1* つの項目：選択することも省略することもできます。

```
[anchovies]
```

- 角カッコと縦線：何も選択しないか、いずれか *1* つだけ選択します。

```
[beans | rice | sweet_potatoes]
```

- 角カッコとカンマ：何も選択しないか、1 つ以上選択します。複数のオプションを選択する場合は、それぞれをカンマで区切ります。

```
[extra_cheese, avocados, sour_cream]
```

反復要素　省略記号 (ピリオド 3 つ) は、最後の要素を必要な回数だけくり返し指定できることを意味します。次の構文では buy が必須のキーワードです。

```
buy thing = price [cash | check | credit]
    [, thing = price [cash | check | credit]]...
```

この例では、製品 (thing) を少なくとも 1 つ購入 (buy) し、価格 (price) を指定する必要があります。支払方法を角カッコの中から 1 つ選択できます。さらに、他の製品を購入することもできます。各 buy に対して、購入した製品 (thing)、価格 (price)、オプションで支払方法 (cash、check、credit のいずれか) を指定します。

アクセシビリティ機能

このマニュアルには、アクセシビリティを重視した HTML 版もあります。この HTML 版マニュアルは、スクリーン・リーダーで読み上げる、または画面を拡大表示するなどの方法により、その内容を理解できるよう配慮されています。

このバージョンの Enhanced Specialty Data Store と HTML マニュアルは、連邦リハビリテーション法第 508 条のアクセシビリティ規定に準拠していることがテストにより確認されています。第 508 条に準拠しているマニュアルは通常、World Wide Web Consortium (W3C) の Web サイト用ガイドラインなど、米国以外のアクセシビリティ・ガイドラインにも準拠しています。

この製品のオンライン・ヘルプは HTML でも提供され、スクリーン・リーダーの読み上げで内容を理解できる機能があります。

注意 アクセシビリティ・ツールを効率的に使用するには、設定が必要な場合もあります。一部のスクリーン・リーダーは、テキストの大文字と小文字を区別して発音します。たとえば、すべて大文字のテキスト (ALL UPPERCASE TEXT など) はイニシャルで発音し、大文字と小文字の混在したテキスト (Mixed Case Text など) は単語として発音します。構文規則を発音するようにツールを設定すると便利かもしれません。詳細については、ツールのマニュアルを参照してください。

Sybase のアクセシビリティに対する取り組みについては、Sybase Accessibility (http://www.sybase.com/accessibility) を参照してください。Sybase Accessibility サイトには、第 508 条と W3C 標準に関する情報へのリンクもあります。

第 1 章

概要

Enhanced Full-Text Search Specialty Data Store (このマニュアルでは拡張型全文検索エンジンと呼ぶ) は、Verity Developer's Kit で利用可能な Verity テクノロジに基づいて構築された Open Server™ アプリケーションです。Adaptive Server は、コンポーネント統合サービス (CIS) を介して拡張型全文検索エンジンに接続し、Verity クエリ言語で記述されたクエリを使用して Adaptive Server データの全文検索を実行できます。

このマニュアルでは、Enhanced Full-Text Search Specialty Data Store の機能について説明します。

拡張型全文検索エンジンの機能

Enhanced Full-Text Search Specialty Data Store 製品は、Adaptive Server データに対して強力な全文検索を実行します。拡張型全文検索エンジンを使用しない Adaptive Server の場合は、select 文の指定と一致したデータのテキスト・カラムのみを検索できます。たとえば、テーブルに犬の種類に関するドキュメントが入っていて、"Saint Bernard" というワードで検索を実行した場合、このクエリではテキスト・カラムに "Saint Bernard" を含むローのみが生成されます。

拡張型全文検索エンジンを使用すると、テキスト・カラムに対するクエリを拡張して、以下の機能を実行できます。

- 選択したドキュメント内に現れる検索項目の頻度に応じて、結果を分類する。たとえば、"Saint Bernard" というワードが 5 回以上使用されているドキュメントのタイトルの一覧を取得できる。
- 検索する複数のワードが *n* ワード以内の近さで現れるドキュメントを選択する。たとえば、"Saint Bernard" と "Swiss Alps" というワードが 10 ワード以内の近さで現れるドキュメントのみを検索できる。
- 1 つの段落または 1 つの文の中に、指定した検索要素をすべて含むドキュメントを選択する。たとえば、"Swiss Alps" というワードが含まれる段落または文の中に "Saint Bernard" というワードも含まれるドキュメントを問い合わせることができる。

- 指定したワードの同義語を 1 つ以上含むドキュメントを選択する。たとえば、“dogs” でドキュメントを選択すると、“dogs”、“canine”、“pooch”、“pup” などのワードを含んだドキュメントが返される。
- カスタム・シソーラスを作成する。たとえば、“Saint Bernard” の同義語として “working dogs”、“St. Bernard”、“large dogs”、“European Breeds” を含んだ、カスタム・シソーラスを作成できる。
- クエリの検索基準を指定するトピックを作成する。たとえば、“Saint Bernard” または “St. Bernard” というフレーズを含むドキュメントを返し、次に “working dogs”、“large dogs”、または “European Breeds” というフレーズを含むドキュメントを返すトピックを作成できる。
- ドキュメントの主要なトピックがわかるように、クラスタにまとめられたドキュメントを返す。
- ドキュメントの中から関連テキストを選択し、他の似たようなドキュメントを検索する。
- Microsoft Word、FrameMaker など、さまざまな種類のドキュメントに対するインデックスを作成する。
- 最大 16 のソート順を使用して、ドキュメントをソートする。
- バックアップおよびリストア機能を統合する。
- システム・プロシージャを使用して設定パラメータの値を変更する。
- サーバがアクティブではないときにテキスト検索用のインデックスを最適化する。これにより、パフォーマンスが向上する。
- 設定情報を表示するための、追加のシステム管理レポートを作成する。
- テキスト検索用にデータベースを自動的にオンラインにする。

高可用性

拡張型全文検索製品は、Sybase フェールオーバをサポートします。Adaptive Server に障害が発生した場合、拡張型全文検索はコンパニオン・サーバからの接続を使用します。また、Adaptive Server にプロキシ・データベースのサポートがある場合、プライマリ・サーバとコンパニオン・サーバの両方が同時に拡張型全文検索を使用できます。

第 2 章

拡張型全文検索エンジンについて

この章では、拡張型全文検索エンジンがどのように機能するかについて説明します。

拡張型全文検索エンジンのコンポーネント

拡張型全文検索エンジンは次のコンポーネントを使用して、全文検索機能を実現します。

- 送信元テーブル
- Verity コレクション (テキスト・インデックス)
- 各種ドキュメント用フィルタ
- text_db データベース
- インデックス・テーブル
- text_events テーブル

送信元テーブル

送信元テーブルは、Adaptive Server によって保持されるユーザ・テーブルです。送信元テーブルには、date、time、text、image、char、varchar、datetime、small datetime、bigint、int、smallint、tinyint、unsigned bigint、unsigned int、unsigned smallint、unitext などのデータ型を使用するカラムが 1 つ以上あり、それらのカラムが、検索されるデータを保持します。送信元テーブルには IDENTITY カラムまたはプライマリ・キーが必要です。IDENTITY カラムは、テキスト検索のときに送信元テーブルをインデックス・テーブルの id カラムとジョインするために使用されます。

送信元テーブルは、実際のデータを保持するローカル・テーブルの場合もあれば、CIS を使用してリモート・データにマップされるプロキシ・テーブルの場合もあります。

Verity コレクション

拡張型全文検索エンジンは、Verity コレクションを使用します。Verity コレクションは、*$SYBASE/$SYBASE_FTS/collections* にあります。「テキスト・インデックスとインデックス・テーブルの作成」(18 ページ) で説明するように、テキスト・インデックスを作成するとき、Verity によって「コレクション」が作成されます。コレクションは、テキスト・インデックスを実現するディレクトリです。拡張型全文検索エンジンは、このコレクションに問い合わせます Verity コレクションの詳細については、Verity Web サイトを参照してください。(http://www.verity.com)

フィルタ

テキスト・インデックスは、フィルタを使用して、ASCII テキストではない、ドキュメント内のタグを取り除きます。拡張型全文検索エンジンは、さまざまなドキュメント・タイプ (Microsoft Word、PDF、WordPerfect、SGML、HTML など) 用のフィルタを備えています。

text_db データベース

「installtextserver スクリプトの実行」(12 ページ) で説明されているように、拡張型全文検索エンジンをインストールするとき、インストール・スクリプト installtextserver によって、text_db という名前のデータベースが Adaptive Server に追加されます。このデータベースは、ユーザ・データを格納せず、vesaux と vesauxcol という 2 つのサポート・テーブルを格納します。これらのテーブルには、Adaptive Server 送信元テーブルと Verity コレクションとの整合性を維持するために拡張型全文検索エンジンが使用するメタデータが入ります。

インデックス・カラムに対して insert、update、または delete が行われた後にコレクションを更新するとき、拡張型全文検索エンジンは vesaux テーブルと vesauxcol テーブルに問い合わせます。これらのテーブルによって、修正されたカラムがどのコレクションにあるかが特定され、影響のあるコレクションがすべて更新されます。またオンラインになったときにも、拡張型全文検索エンジンはこれらのテーブルを使用して必要なコレクションがすべてあるかを確認します。

vesaux テーブル

vesaux テーブルのカラムを表 2-1 に示します。

表 *2-1: vesaux* テーブルのカラム

カラム名	説明
id	IDENTITY カラム
object_name	外部インデックスが作成される送信元テーブルの名前
option_string	テキスト・インデックス作成オプション
collection_id	Verity コレクションの名前
key_column	送信元テーブルの IDENTITY カラムまたはプライマリ・キーの名前
svrid	コレクションを保持している拡張型全文検索エンジンのサーバ ID

vesauxcol テーブル

vesauxcol テーブルのカラムを表 2-2 に示します。

表 *2-2: vesauxcol* テーブルのカラム

カラム名	説明
id	参照されるローの、vesaux テーブルにおける ID
col_name	検索するカラムの名前
col_type	カラムのタイプ (date、time、text、image、char、varchar、datetime、smalldatetime。拡張型全文検索エンジンの場合は、int、smallint、tinyint も可能)

インデックス・テーブル

インデックス・テーブルを使用すると、送信元テーブルに格納されているドキュメントを検索することができます。**インデックス**・テーブルは、拡張型全文検索エンジンによって管理され、対応する送信元テーブルの IDENTITY カラムまたはプライマリ・キーにマップする id カラムを持ちます。送信元テーブルのローの IDENTITY またはプライマリ・キーの値は、データとともに Verity コレクションに格納されるので、送信元テーブルとインデックス・テーブルのジョインが可能です。インデックス・テーブルは拡張型全文検索エンジンによって格納、管理されますが、コンポーネント統合サービスを通じて Adaptive Server にとってのプロキシ・テーブルとして機能します。

インデックス・テーブルには疑似カラムと呼ばれる特別なカラムがあります。疑似カラムは拡張型全文検索エンジンによって、検索パラメータと送信元テーブル内のテキスト・データのロケーションを決定するために使用されます。疑似カラムには、対応する物理格納領域がありません。つまり、疑似カラムの値はクエリの間だけ有効で、クエリの実行が終了するとすぐに削除されます。

たとえば、クエリで score 疑似カラムを使用するとき、クエリと一致する程度に従って各ドキュメントをランク付けするため、テキスト・データベース内で “small Saint Bernards” というフレーズを見つけるには、*score* に 15 を使用することになるかもしれません。このフレーズはあまり頻繁に発生しないため、小さな score 値を使用することで、検索基準の発生回数が少ないドキュメントが含まれるように検索範囲を広げます。しかし、“large Saint Bernards” のように一般的なフレーズを検索している場合は、*score* に 90 を使用して、検索基準の発生回数が多いドキュメントに検索を限定することができます。

検索に含めるパラメータを指定するには、score カラムと他の疑似カラム (id、index_any、sort_by、summary、max_docs) を使用します。疑似カラムについては、「インデックス・テーブルの疑似カラム」(42 ページ) を参照してください。

text_events テーブル

テキスト・インデックスがあるテーブルを持つ各データベースには、インデックス・カラムへの挿入、更新、削除のログを取る「イベント・テーブル」が必要です。このテーブルの名前は、text_events で、このテーブルは、更新されたデータを Verity コレクションに伝達するために使用されます。

表 2-3 は、text_events テーブルのカラムです。

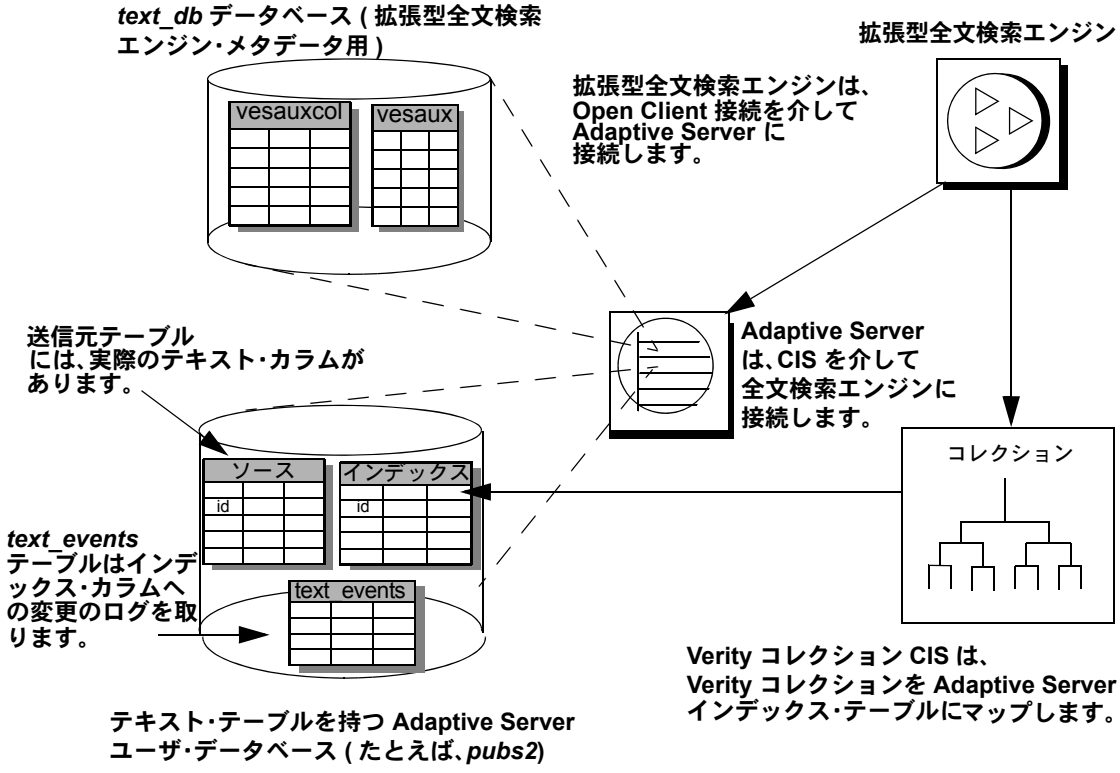
表 *2-3: text_events* テーブルのカラム

カラム名	説明
event_id	IDENTITY カラム
id	更新、挿入、削除されたローの ID
tableid	更新、挿入、削除されたローがあるテーブルの名前
columnid	text インデックスが作成されたカラムの名前
event_date	update、insert、または delete の日付と時間
event_type	更新の種類 (update、insert、または delete)
event_status	update、insert、または delete がコレクションに伝達されたかどうかを示す。 • 0 − イベント「未読」 • 1 − イベント「読み込み済」 • 2 − イベント「成功」 • 3 − イベント「失敗」
srvid	コレクションを保持している拡張型全文検索エンジンのサーバ ID

コンポーネント間の関係

拡張型全文検索エンジンのコンポーネント間の関係を、図 2-1 に示します。

図 *2-1:* 拡張型全文検索エンジンのコンポーネント



全文検索の動作の仕組み

全文検索を実行するには、送信元テーブルの IDENTITY カラムまたはプライマリ・キーとインデックス・テーブルの id カラムをジョインする select 文を入力します。検索を定義するために、必要に応じて疑似カラムも使用します。たとえば、次のクエリは、“Greek” というワードが “Gustibus” というワードの近くにある、pubs2 データベースの blurbs テーブル内のドキュメントを検索します (i_blurbs テーブルはインデックス・テーブルです)。

```
select t1.score, t2.copy
from i_blurbs t1, blurbs t2
where t1.id=t2.id and t1.score > 20
and t1.max_docs = 10
and t1.index_any = "<near>(Greek, Gustibus)"
```

Adaptive Server と拡張型全文検索エンジンは、次のようにクエリ処理を分担しています。

1 拡張型全文検索エンジンがクエリを次のように処理します。

```
select t1.score, t1.id
from i_blurbs t1
where t1.score > 20
and t1.max_docs = 10
and t1.index_any = "<near>(Greek, Gustibus)"
```

select 文に、Verity 演算子 near と、疑似カラム score、max_docs、index_any が含まれています。これらの演算子と疑似カラムは Verity コレクションに対する検索用のパラメータになり、これらにより、copy カラム全体からの結果セットは、“Greek” と “Gustibus” が互いに近い上位 10 個のドキュメントまで絞られます。

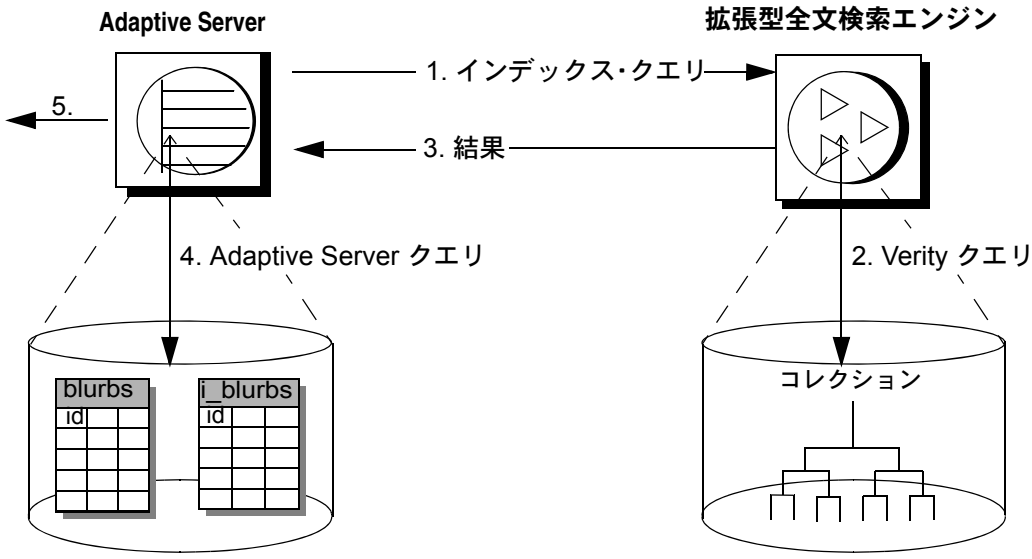
2 Adaptive Server は、拡張型全文検索エンジンが手順 1 で返した結果セットに対して、次の select 文を処理します。

```
select t1.score, t2.copy
from i_blurbs t1, blurbs t2
where t1.id=t2.id
```

これにより、blurbs テーブルと i_blurbs テーブル (それぞれ送信元テーブルとインデックス・テーブル) が、blurbs テーブルの IDENTITY カラムまたはプライマリ・キーと i_blurbs テーブルの id カラムでジョインされます。

図 2-2 は、Adaptive Server と拡張型全文検索エンジンがクエリを処理する方法を示します。

図 2-2: 全文検索クエリの処理



1. **Adaptive Server が拡張型全文検索エンジンにインデックス・クエリを送信する**
2. **拡張型全文検索エンジンがクエリ内の Verity 演算子を処理し、コレクションから結果セットを生成する**
3. **拡張型全文検索エンジンが Adaptive Server に結果セットを返す**
4. **Adaptive Server がローカル・テーブルで select 文を処理する**
5. **Adaptive Server がクエリの結果を紛示する**

第 3 章

Adaptive Server の全文検索の設定

この章では、全文検索を実行するように Adaptive Server を設定する方法について説明します。

Adaptive Server の拡張型全文検索エンジン用の設定

拡張型全文検索エンジンは、Adaptive Server が CIS (コンポーネント統合サービス) を介して接続するリモート・サーバです。Adaptive Server を次のように設定すると、拡張型全文検索エンジンを使用できるようになります。

- enable cis、cis rpc handling、full-text search 設定パラメータが有効になっていない場合は、有効にします。full-text search を有効にするにはライセンスが必要です。
- installtextserver スクリプトを実行して、拡張型全文検索エンジンを 1 つ以上定義します。
- installmessages スクリプトを実行して、拡張型全文検索エンジンのシステム・プロシージャ用のメッセージをインストールします。
- installevent スクリプトを実行して、テキスト・インデックスを保存するユーザ・データベースごとに text_events テーブルを作成します。
- ローカル・サーバを指定して再起動します。

設定パラメータの有効化

拡張型全文検索エンジンに接続するには、enable cis 設定パラメータと cis rpc handling 設定パラメータが有効になった状態で Adaptive Server が稼働している必要があります。これらのパラメータが有効でない場合には、isql を使用して Adaptive Server にログインし、sp_configure を使用して有効にします。次に例を示します。

```
exec sp_configure "enable cis", 1
exec sp_configure "cis rpc handling", 1
exec sp_configure "enable full-text search", 1
```

enable cis を変更した場合には、新しい設定パラメータを有効にするために Adaptive Server を再起動する必要があることを示すメッセージが表示されます。

installtextserver スクリプトの実行

installtextserver スクリプトは、次のことを行います。

- 拡張型全文検索エンジンを、サーバ・クラス sds のリモート・サーバとして Adaptive Server に対して定義する。
- テキスト・インデックス・メタデータを格納するデータベースを作成する。このデータベースの詳細については、「text_db データベース」(4 ページ) を参照のこと。
- 拡張型全文検索エンジンが必要とするシステム・プロシージャをインストールする。

installtextserver スクリプトを 1 回だけ実行します (「installtextserver スクリプトの起動」(14 ページ) を参照してください)。後から拡張型全文検索エンジンを追加するには、sp_addserver を使用します。sp_addserver の詳細については、「複数の拡張型全文検索エンジンの設定」(79 ページ) を参照してください。

注意 *installtextserver* スクリプトを model データベースに対して実行する場合は、スクリプトを実行する前に、model データベースのサイズを少なくとも 3MB に増やしてください。

すべての拡張型全文検索エンジンは、テキスト・インデックス・メタデータの格納に同じデータベースを使用します。このデータベースは、デフォルトの text_db データベースと呼ばれています。

installtextserver スクリプトによって追加されるシステム・プロシージャの一覧と説明については、「付録 A システム・プロシージャ」 を参照してください。

installtextserver スクリプトの編集

installtextserver スクリプトは *$SYBASE/$SYBASE_FTS/scripts* ディレクトリにあります。テキスト・エディタ (vi や emacs など) を使用し、このスクリプトを開いて編集します。この編集では、次のことができます。

- text_db データベースの名前を変更すること。別の名前を使用する場合は、すべての text_db を、該当する名前に変更してください。

  注意　*text_db* データベースの名前を変更する場合は、defaultDb 設定パラメータ内の名前も変更する必要があります (「設定パラメータの修正」(62 ページ) を参照してください)。

- 拡張型全文検索エンジンの名前を変更すること。デフォルトでは、installtextserver スクリプトによって、“textsvr” という名前の拡張型全文検索エンジンが定義されています。使用する拡張型全文検索エンジンの名前が異なる場合は、このスクリプトを編集して適切なサーバ名を定義してください。

- 複数の拡張型全文検索エンジンを追加すること (これによってパフォーマンスがどのように向上するかについては、「複数の拡張型全文検索エンジンの設定」(79 ページ) を参照してください)。初期設定として複数の拡張型全文検索エンジンを定義する場合は、installtextserver スクリプトを編集して、すべての拡張型全文検索エンジンの定義が含まれるようにしてください。installtextserver には、設定する拡張型全文検索エンジンを指定するための次のセクションがあります (デフォルトは “textsvr” です)。

  ```
  /*
  ** Add the text server
  */
  exec sp_addserver textsvr,sds,textsvr
  go
  ```

  設定する拡張型全文検索エンジンそれぞれに対するエントリを追加してください。たとえば、KRAZYKAT、OFFICAPUP、MOUSE という名前の 3 つの拡張型全文検索エンジンを設定する場合、デフォルトの “textsvr” 行を次の行で置き換えます。

  ```
  exec sp_addserver KRAZYKAT, sds, KRAZYKAT
  exec sp_addserver OFFICAPUP, sds, OFFICAPUP
  exec sp_addserver MOUSE, sds, MOUSE
  go
  ```

- 拡張型全文検索エンジンとの接続に Enterprise Connect Data Access を使用する場合に、vesaux テーブルと vesauxcol テーブルの sp_addobjectdef コールのサーバ名指定を有効なリモート・サーバに変更すること。たとえば、リモート・サーバの名前が REMOTE の場合は、次の行を変更します。

  ```
  exec sp_addobjectdef
  "vesaux","SYBASE.master.dbo.vesaux","table"
  exec sp_addobjectdef
  "vesauxcol","SYBASE.master.dbo.vesauxcol",
  "table"
  ```

  上記を次のように変更します。

  ```
  exec sp_addobjectdef
  "vesaux","REMOTE.master.dbo.vesaux","table"
  exec sp_addobjectdef
  "vesauxcol","REMOTE.master.dbo.vesauxcol",
  "table"
  ```

installtextserver スクリプトの起動

isql を使用して、installtextserver スクリプトを実行します。たとえば、MYSVR という名前の Adaptive Server で installtextserver スクリプトを実行するには、次のように入力します。

```
isql -Usa -P -SMYSVR -i
$SYBASE/$SYBASE_FTS/scripts/installtextserver
```

installmessages スクリプトの実行

拡張型全文検索エンジンには、Adaptive Server にインストールする必要がある独自のシステム・プロシージャ・メッセージがあります。installmessages スクリプトを使用して、それらのメッセージをインストールします。拡張型全文検索エンジンが複数ある場合でも、installmessages スクリプトの実行は 1 回だけで済みます。

たとえば、MYSVR という名前のサーバで installmessages スクリプトを実行するには、次のように入力します。

```
isql -Usa -P -SMYSVR -i
$SYBASE/$SYBASE_FTS/scripts/installmessages
```

installevent スクリプトの実行

text インデックスによって参照されるテーブルを持つ各データベースには、インデックス・カラムへの挿入、更新、削除のログを取る text_events テーブルが必要です。このテーブルは、更新されたデータを Verity コレクションに伝達するために使用されます。

次に説明するように installevent スクリプトを実行して、text_events テーブルと、関連するシステム・プロシージャをデータベース内に作成します。次の手順で installevent スクリプトを使用します。

- すべてのデータベースでテキスト・インデックスが必要な場合は、installevent スクリプトを実行して、model データベースに text_events テーブルを作成する。新しく作成されたデータベースごとに、text_events テーブルを作成する。text_events テーブルを既存のデータベースに追加するには、次に説明するようにスクリプトを編集し、既存のユーザ・データベースに text_events テーブルを作成する。
- すべてのデータベースでテキスト・インデックスが必要ではない場合は、installevent スクリプトをサンプルとして使用する。テキスト・インデックスが必要なテーブルを持つ既存のデータベースと新しいデータベースのそれぞれに対して、installevent スクリプトを実行する。ただし、次で説明するようにこのスクリプトを編集して、適切なユーザ・データベースに text_events テーブルが作成されるようにする必要がある。

注意　テキスト・インデックスが必要な送信元テーブルを持つデータベースに text_events テーブルが存在しない場合は、送信元テーブルに対する変更は Verity コレクションに伝達されません。

installevent スクリプトの編集

installevent スクリプトは、*$SYBASE/$SYBASE_FTS/scripts* ディレクトリにあります。テキスト・エディタ (vi や emacs など) を使用し、このスクリプトを開いて編集します。この編集では、次のことができます。

- ユーザ・データベース名を変更すること。installevent スクリプトにより、model データベースに (text_events という名前の) イベント・テーブルと、関連するシステム・プロシージャが作成されます。model データベースはデフォルトのデータベースです。既存のユーザ・データベースに text_events テーブルをインストールするには、スクリプトを編集して、model への参照をすべてユーザ・データベース名に置き換えてください。

- text_db データベース名を変更すること。テキスト・インデックス・メタデータを格納するデータベースの名前が text_db 以外の場合は、text_db への参照をすべて適切な名前に置き換えてください。

注意 text_db データベースの名前は、defaultDb 設定パラメータ内の名前と同じである必要があります (「設定パラメータの修正」(62 ページ) を参照してください)。

installevent スクリプトの実行

isql を使用して installevent スクリプトを実行し、text_events テーブルと、関連するシステム・プロシージャを Adaptive Server にインストールします。たとえば、MYSVR という名前のサーバで installevent スクリプトを実行するには、次のように入力します。

```
isql -Usa -P -SMYSVR -i
$SYBASE/$SYBASE_FTS/scripts/installevent
```

注意 installevent スクリプトを実行するには、text_db データベースが存在している必要があります。存在しない場合は、先に installtextserver スクリプトを実行します。

注意 64 ビット・プラットフォームに EFTS-12_5 スクリプト (*installtextserver*、*installevents*、*installmessages*) をインストールする場合は、事前に tempdb データベースと model データベースのサイズをそれぞれデフォルト・サイズから 3MB に増やしてください。

ローカルの Adaptive Server の指定

Adaptive Server 12.5 以降で拡張型全文検索エンジンを使用するときには、sp_addserver <servername>, local を使用してローカルの Adaptive Server を指定します。sp_addserver を発行したあと、ローカルの Adaptive Server を再起動します。システム・ストアド・プロシージャは、model データベースにはインストールせず、sybsystemprocs データベースにインストールしてください。

テキスト・インデックスの作成と管理

ユーザ・データベース内の送信元テーブルに対してテキスト・インデックスを作成すると、拡張型全文検索エンジンは全文検索を処理できるようになります。テキスト・インデックスの作成後は、送信元データが変わるときにテキスト・インデックスを更新して、最新に保つ必要があります。テキスト・インデックスを作成し、管理するには、次の手順に従います。

1　インデックスを作成するための送信元テーブルを設定する (「送信元テーブルのインデックスの設定」(17 ページ) を参照してください)。

2　テキスト・インデックスとインデックス・テーブルを作成する (「テキスト・インデックスとインデックス・テーブルの作成」(18 ページ) を参照してください)。

3　全文検索のためにデータベースをオンラインにする (「全文検索のためにデータベースをオンラインにする」(21 ページ) を参照してください)。

4　ユーザ・データ内の変更をテキスト・インデックスに伝達する (「テキスト・インデックスに対する変更の伝達」(21 ページ) を参照してください)。

5　テキスト・インデックスをレプリケートする場合は、送信先データベースでテキスト・インデックスを設定する (「テキスト・インデックスのレプリケート」(22 ページ) を参照してください)。

テキスト・インデックスの設定例については、*$SYBASE/$SYBASE_FTS/sample/scripts* ディレクトリのサンプル・スクリプト sample_text_main.sql を参照してください。

送信元テーブルのインデックスの設定

送信元テーブルには、検索の実行対象のデータがあります (たとえば、pubs2 データベースの blurbs テーブル)。詳細については、「送信元テーブル」(3 ページ) を参照してください。

送信元テーブルにテキスト・インデックスを作成する前に、次のことを行います。

- 送信元テーブルに IDENTITY カラムまたはプライマリ・キーがあることを確認する。存在しない場合は、テーブルを変更し、IDENTITY カラムを追加する。
- プライマリ・キーまたは IDENTITY カラムにユニーク・インデックスを作成する (オプション)。

送信元テーブルには、各ローをユニークに識別し、インデックス・テーブルと送信元テーブルをジョインするために使用される IDENTITY カラムまたはプライマリ・キーが必要です。テキスト・インデックスを作成するときに、IDENTITY カラムまたはプライマリ・キーはインデックス・カラムとともに拡張型全文検索エンジンに渡されます。IDENTITY カラムまたはプライマリ・キーの値はテキスト・インデックスに格納され、インデックス・テーブルの id カラムにマップされます。プライマリ・キーを持たないテーブルの場合は、IDENTITY カラムをテーブルに追加できます。

送信元テーブルへの IDENTITY カラムの追加

composers という名前のテーブルに IDENTITY カラムを作成するには、次のようにテーブルを定義します。

```
create table composers (
    id          numeric(m,n)     identity,
    comp_fname  char(30)         not null,
    comp_lname  char(30)         not null,
    text_col    text
)
```

このとき、m =< 38 で、常に n = 0 です。

既存のテーブルに IDENTITY カラムを追加するには、次の構文を使用します。

```
alter table table_name add id numeric(10,0) identity
```

IDENTITY カラムへのユニーク・インデックスの追加

最適なパフォーマンスを得るために、Sybase では IDENTITY カラムに対してユニーク・インデックスを作成することをおすすめします。たとえば、先程作成した IDENTITY カラムに comp_id という名前のユニーク・インデックスを作成するには、次のように入力します。

```
create unique index comp_id
on composers(id)
```

ユニーク・インデックスの作成の詳細については、『Transact-SQL ユーザーズ・ガイド』の「第 11 章 テーブルのインデックスの作成」を参照してください。

テキスト・インデックスとインデックス・テーブルの作成

sp_create_text_index を使用して、テキスト・インデックスを作成します。sp_create_text_index は、次のことを実行します。

- text_db データベースの vesaux テーブルと vesauxcol テーブルの更新
- テキスト・インデックス (Verity コレクション) の作成
- Verity コレクションの移植

- 送信元テーブルがあるユーザ・データベースへのインデックス・テーブルの作成

注意 sp_create_text_index を正常に実行するには、拡張型全文検索エンジンが稼働中である必要があります。拡張型全文検索エンジンの起動と停止については、「第 6 章 システム管理」を参照してください。

テキスト・インデックスのカラム数は 16 個までです。インデックスを作成できるカラムのデータ型は、char、varchar、nchar、nvarchar、date、time、text、image、datetime、smalldatetime、int、smallint、tinyint、unichar、univarchar です。

たとえば、KRAZYKAT の pubs2 にある、blurbs テーブルの copy カラムに対して、i_blurbs という名前のテキスト・インデックスとインデックス・テーブルを作成するには、次のように入力します。

```
sp_create_text_index "KRAZYKAT", "i_blurbs", "blurbs", " ", "copy"
```

パラメータの意味：

- KRAZYKAT は、拡張型全文検索エンジンの名前です。
- i_blurbs は、作成するインデックス・テーブルとテキスト・インデックスの名前です。
- blurbs は、テキスト・インデックスを作成する送信元テーブルです。
- " " は、テキスト・インデックス作成オプション用のプレースホルダです。
- copy は、インデックスを作成する blurbs テーブル内のカラムです。

詳細については、sp_create_text_index (133 ページ) を参照してください。

注意 設定ファイルの text_db データベース名 (defaultDb パラメータの次に記述されています) は、Adaptive Server のデータベース名と必ず一致させてください。一致しない場合、テキスト・インデックスを作成できません。また、text_events テーブルが、ユーザ・データベースに存在することも確認してください。存在しない場合は、そのデータベースに対して installevent スクリプトを実行してください (「installevent スクリプトの実行」(15 ページ) を参照してください)。

インデックスを作成するデータの量によって、Verity コレクションの移植には数分または数時間かかる場合があります。この手順は、サーバの稼働率が低いときに実行した方が効率的です。batch_size 設定パラメータの値を大きくしても、処理は効率化されます。詳細については、「batch_size」(78 ページ) を参照してください。

注意 インデックスの名前は変更しないでください。これは、Verity コレクションの名前が変更されないからです。

テキスト・インデックス作成時の複数カラムの指定

複数のカラムにテキスト・インデックスを作成すると、テキスト・インデックス内のカラムは、カラムごとのドキュメント・ゾーンに置かれます。ゾーンの名前は、カラムの名前と同じです。たとえば、KRAZYKAT の pubs2 にある、blurbs テーブルの copy カラムと au_id カラムの両方に対して、i_blurbs という名前のテキスト・インデックスとインデックス・テーブルを作成するには、次のように入力します。

```
sp_create_text_index "KRAZYKAT", "i_blurbs", "blurbs", " ", "copy", "au_id"
```

sp_create_text_index によって、テキスト・インデックス内に “copy” と “au_id” という名前の 2 つのゾーンが作成されます。i_blurbs テキスト・インデックスに対してクエリを発行すると、検索には copy および au_id カラムが含まれます。ただし、in 演算子でドキュメント・ゾーンを指定することによって、検索を特定のカラムに限定できます (詳細については、「in」(51 ページ) を参照してください)。

テキスト・インデックス・プロキシ・テーブルに対するパーミッションの付与

sp_create_text_index によってテキスト・インデックスが作成された後、管理者は、そのテキスト・インデックスに対して問い合わせを実行するユーザまたはグループに、テキスト・インデックス・プロキシ・テーブルに対する select パーミッションを明示的に付与する必要があります。

ユーザにテキスト・インデックスに対する select パーミッションを付与するには、そのユーザがテキスト・インデックス・ソース内のすべてのカラムに対する select パーミッションを持っている必要があります。ユーザがテキスト・インデックス・ソース内のどのカラムに対する select パーミッションも持っていない場合、管理者はテキスト・インデックスに対する select パーミッションを付与できません。

テキスト・インデックス・ソース・テーブル内のいずれかのカラムが暗号化カラムの場合、管理者は、暗号化カラムに対する decrypt パーミッションを持つユーザに対してのみ、テキスト・インデックスに対する select パーミッションを付与する必要があります。ユーザが暗号化カラムに対する select パーミッションを持っていても、そのユーザのパーミッションが nodecrypt である場合、管理者はテキスト・インデックスに対する select パーミッションを付与してはなりません。

全文検索のためにデータベースをオンラインにする

拡張型全文検索エンジンでは、auto_online 設定パラメータを 1 に設定すると、データベースは自動的にオンラインになります。

データベースをオンラインにすると、拡張型全文検索エンジンによって Verity の内部構造体が初期化され、Verity コレクションが存在することが確認されます。

全文検索のために、自動的にオンラインにならない場合にデータベースをオンラインにするには、sp_text_online を使用します。たとえば、KRAZYKAT という名前の拡張型全文検索エンジンで blurbs テーブルに対する全文検索を発行するために pubs2 データベースをオンラインするには、次のように入力します。

```
sp_text_online KRAZYKAT, pubs2
```

次のメッセージが表示されます。

```
Database 'pubs2' is now online
```

これで、pubs2 データベースに対して全文検索が実行できるようになりました。

詳細については、sp_text_online (149 ページ) を参照してください。

テキスト・インデックスに対する変更の伝達

送信元テーブルのデータに対して、insert、update、または delete を実行しても、テキスト・インデックスは自動的には更新されません。データの更新後に、sp_refresh_text_index を実行して、text_events テーブルに変更内容のログを取ります。次に、sp_text_notify を実行して、変更を処理する必要があることを拡張型全文検索エンジンに通知します。拡張型全文検索エンジンは Adaptive Server に接続し、text_events テーブルのエントリを読み込んで、どのインデックス、テーブル、ローに影響があるかを判断し、適切なコレクションを更新します。

これらのシステム・プロシージャの詳細については、sp_refresh_text_index (139 ページ) および sp_text_notify (148 ページ) を参照してください。

各 insert、update、または delete の後で sp_refresh_text_index を自動的に実行するには、次のように、送信元テーブルにトリガを作成します。

- delete オペレーションの後に、sp_refresh_text_index を実行するトリガを作成する。
- insert オペレーションの後に、sp_refresh_text_index を実行するトリガを作成する。
- インデックス・カラムへの update オペレーションの後に、sp_refresh_text_index を実行するトリガを作成する。

writetext を使用して text カラムを更新するときには、トリガは起動されません。writetext の後で sp_refresh_text_index を自動的に実行するには、次の手順に従います。

- 非 text カラムを設定し、各 writetext の実行後にそのカラムの update を行う。
- その非 text カラムに、sp_refresh_text_index を実行するトリガを作成する。sp_text_notify を発行したときに、拡張型全文検索エンジンはロー全体を再び挿入するので、text カラムへの更新はテキスト・インデックスに伝達される。

これらの各トリガの例については、*$SYBASE/$SYBASE_FTS/sample/scripts* ディレクトリのサンプル・スクリプト sample_text_main.sql を参照してください。

テキスト・インデックスのレプリケート

テキスト・インデックスを持ったテーブルをレプリケートするには、次のガイドラインに従います。

- 送信先データベースにテーブル定義を作成する。
- text_events テーブルがまだ存在していない場合は、installevent スクリプトを実行して、送信先データベースに text_events テーブルを作成する (「installevent スクリプトの実行」(15 ページ) を参照してください)。
- sp_create_text_index を実行して、送信先データベースの空のテーブルにテキスト・インデックスを作成する (「テキスト・インデックスとインデックス・テーブルの作成」(18 ページ) を参照してください)。
- text_events テーブルのデータに対して insert、update、または delete を実行するときには必ず、このテーブルにエントリを insert するために sp_create_text_index を実行するトリガを作成する (「テキスト・インデックスに対する変更の伝達」(21 ページ) を参照してください)。
- Replication Server にレプリケーション定義を作成する。これにより、送信元テーブルのすべてのデータが送信先テーブルにレプリケートされる。詳細については、『Replication Server 管理ガイド』を参照してください。
- sp_text_notify を実行して、テキスト・インデックスの update を行う。すなわち、sp_text_notify を定期的に実行して、送信先テーブルへの変更を処理する (「テキスト・インデックスに対する変更の伝達」(21 ページ) を参照してください)。

注意 writetext コマンドを実行するときは常に、非 text カラムに対して update を発行する必要があります。これによって、text_events テーブルにデータを挿入するトリガが起動されます。

例：新しいデータベースのテキスト検索の有効化

この例では、movies データベース内の reviews テーブルの plot カラムに、text インデックスを作成する手順について説明します。この処理では、次のことを前提にしています。

- MYSVR サーバの movies という新しいデータベースに reviews テーブルが作成されていること
- reviews テーブルに、これからインデックスを作成する plot という名前のカラムがあること
- Adaptive Server と、MYTXTSVR という名前の拡張型全文検索エンジンが互いに接続するように設定されていること

手順 1. *text_events* テーブルが存在することを確認する

text インデックスによって参照されるテーブルを持つ各データベースには、インデックス・カラムへの挿入、更新、削除のログを取る text_events テーブルが必要です。

text_events テーブルが model データベースにある場合、すべての新しいデータベースにも text_events テーブルがあります。text_events テーブルが model データベースにない場合は、installevent スクリプトを実行して新しいデータベースに text_events テーブルをインストールします。たとえば、movies データベースに text_events テーブルをインストールするには、次の手順に従います。

- installevent スクリプトを installeventmovies として保存する
- スクリプトを編集して、*model* というワードを使用している箇所をすべて movies というワードで置き換える
- 次のスクリプトを実行する

```
isql -Usa -P -SMYSVR -i
$SYBASE/$SYBASE_FTS/scripts/installeventmovies
```

text_events テーブルのインストールの詳細については、「installevent スクリプトの実行」(15 ページ) を参照してください。

手順 2. IDENTITY カラムまたはプライマリ・キーをチェックする

送信元テーブルには、各ローをユニークに識別し、インデックス・テーブルと送信元テーブルをジョインするために使用される IDENTITY カラムまたはプライマリ・キーが必要です。

たとえば、reviews テーブルに IDENTITY カラムを追加するには、次のように入力します。

```
alter table reviews add id numeric(10,0) identity
```

詳細については、「送信元テーブルへの IDENTITY カラムの追加」(18 ページ) を参照してください。

手順 3. IDENTITY カラムにユニーク・インデックスを作成する

この手順は任意です。パフォーマンスを向上させるために、Sybase では IDENTITY カラムのみを持ったユニーク・インデックスを作成することをおすすめします。たとえば、前の手順で作成した IDENTITY カラムに reviews_id という名前のユニーク・インデックスを作成するには、次のコマンドを発行します。

```
create unique index reviews_id on reviews(id)
```

ユニーク・インデックスの作成の詳細については、『Transact-SQL ユーザーズ・ガイド』の「第 11 章 テーブルのインデックスの作成」を参照してください。

手順 4. テキスト・インデックスとインデックス・テーブルを作成する

ユーザ・データベースの送信元テーブルには、全文検索が実行できるようにインデックスを作成します。たとえば、reviews テーブルの plot カラムに、reviews_idx という名前のテキスト・インデックスとインデックス・テーブルを作成するには、次のように入力します。

```
sp_create_text_index "MYTXTSVR", "reviews_idx",
"reviews", " ", "plot"
```

これで、reviews テーブルに対して全文検索が実行できるようになりました。

詳細については、sp_create_text_index (133 ページ) を参照してください。

手順 5. 全文検索のためにデータベースをオンラインにする

MYTXTSVR という名前の拡張型全文検索エンジンのために movies データベースをオンラインにするには、次のように入力します。

```
sp_text_online MYTXTSVR, movies
```

注意 auto_online が “1” に設定されている場合には、この手順は省略してください。

詳細については、sp_text_online (149 ページ) を参照してください。

ユーロ記号のインデックス作成

次の設定のガイドラインに従っている場合、ユーロ記号のインデックスを作成し、ユーロ記号を適切に返すことができます。Adaptive Server に utf8 文字セットがインストールされている必要があります。拡張型全文検索で、vdkLanguage を <language>x に設定し、vdkCharset をブランクにします。次に例を示します。

```
ASE 12.5.x charset = utf8
EFTS 12.5.x vdkLanguage = englishx
EFTS 12.5.x vdkCharset =
```

第 4 章

Verity 機能の設定

この章では、Verity 機能を使用してクエリを作成できるように、必要な設定について説明します。

QBE (例示による問い合わせ)、要約、クラスタリングの有効化

style.prm ファイルでテキスト・インデックスに含める追加データを指定して、以下の機能をサポートします。

- QBE − あるフレーズに似たドキュメントの検索 (詳細については、「like」(51 ページ) を参照してください)。

  注意 テキスト・インデックスは、like 演算子の QBE フォームでフレーズをサポートする追加データを必要とします。QBE フォームでドキュメントを使用する場合は、追加データを必要としません。

- 要約 − ドキュメント全体ではなく、ドキュメントの要約を返す (詳細については、「summary カラムを使用したドキュメントの要約」(45 ページ) を参照してください)。
- クラスタリング − サブトピックを使用して、結果セット内のドキュメントをグループ化する。(詳細については、「疑似カラムを使用した、クラスタード結果セットの要求」(46 ページ) を参照してください)。

以上の 3 つの機能については、マスタ *style.prm* ファイルを編集してすべてのテキスト・インデックスで有効にしたり、テキスト・インデックスごとに *style.prm* ファイルを編集して個別に有効にしたりすることができます。これら方法については、後述します。

QBE とクラスタリング

QBE フォームにフレーズを指定したり、クラスタリングを使用したりするには、インデックス作成時にドキュメント機能ベクトルの生成を有効にしてください。これを実行するには、*style.prm* ファイルの次の行をコメント解除します。

```
$define DOC-FEATURES "TF"
```

要約

要約を使用できるように拡張型全文検索エンジンを設定するには、*style.prm* ファイル内の “#$define” で始まる次の行のうちの 1 つをコメント解除します。

```
# The example below stores the best three sentences of
# the document, but not more than 255 bytes.
#$define DOC-SUMMARIES   "XS MaxSents 3 MaxBytes 255"
# The example below stores the first four sentences of
# the document, but not more than 255 bytes.
#$define DOC-SUMMARIES   "LS MaxSents 4 MaxBytes 255"
# The example below stores the first 150 bytes of
# the document, with whitespace compressed.
#$define DOC-SUMMARIES   "LB MaxBytes 150"
```

上記は、行ごとに要約のレベルが異なります。“#$define” 行の最後に指定されている数字を変更して、拡張型全文検索エンジンで表示するデータのバイト数を指定できます。たとえば、データの先頭から 233 バイト目までの要約を行うには、スクリプトを次のように編集します。

```
$define DOC-SUMMARIES   "LS MaxSents 4 MaxBytes 233"
```

最大で 255 バイトまで表示できます。256 以上の値を指定しても、255 にトランケートされます。

マスタ *style.prm* ファイルの編集

注意 マスタ *style.prm* ファイルのロケーションは、EFTS 12.5.2 リリースから変更されています。*$SYBASE/$SYBASE_FTS/verity/common/style* にある *style.prm* ファイルに対して行われる編集は無視されます。新しいロケーションについては以下で説明します。

マスタ *style.prm* ファイルは、*$SYBASE/$SYBASE_FTS/verity/common/styles/txtsvr* にあります。このファイルには、拡張型全文検索エンジンのデフォルトのスタイル・パラメータが入っています。どのテーブルでインデックスを作成しても、クラスタリング、QBE フォームでのリテラル・テキストの指定、または要約ができるように拡張型全文検索エンジンを設定するには、このファイルを編集します。前述のように、使用する行のコメントを解除してください。

注意 既存のテキスト・インデックスがある場合は、「個々の style.prm ファイルの編集」(27 ページ) で説明する手順に従って、前述の機能が使用できるようにテキスト・インデックスを再作成してください。

個々の *style.prm* ファイルの編集

次の手順を実行して拡張型全文検索エンジンを設定し、個々のテキスト・インデックスで、クラスタリング、QBE フォームでのリテラル・テキストの指定、または要約ができるようにします。

1　sp_create_text_index を使用して、テキスト・インデックスを作成します。*option_string* パラメータには “empty” を使用します。これは、テキスト・インデックス用の *style.prm* ファイルは作成しても、Verity コレクションにデータを移植しないようにするためです。たとえば、blurbs テーブルの copy カラムのクラスタリングを有効にする場合は、次の構文を使用します。

```
sp_create_text_index "KRAZYKAT", "i_blurbs", "blurbs", "empty", "copy"
```

注意 既存のテキスト・インデックスがある場合は、この手順を省略します。テキスト・インデックスを再作成する必要はありません。

2　sp_drop_text_index を使用して、編集する *style.prm* ファイルに関連付けられたテキスト・インデックスを削除します。

たとえば、手順 1 で作成したテキスト・インデックスを削除するには、次のように入力します。

```
sp_drop_text_index "blurbs.i_blurbs"
```

3　テキスト・インデックス用の *style.prm* ファイルを編集します。既存のコレクションの *style.prm* ファイルは、*$SYBASE/$SYBASE_FTS/collections/db.owner.index/style* にあります。

たとえば、pubs2 データベースに i_blurbs というテキスト・インデックスを作成した場合、このファイルへのフル・パスは *$SYBASE/$SYBASE_FTS/collections/pubs2.dbo.i_blurbs/style* になります。ここで、*db.owner.index* は、データベース、データベース所有者、および sp_create_text_index を使用して作成したインデックスを表します。

4　前述のように、使用する行のコメントを解除してください。たとえば、クラスタリングを有効にする場合は、次の行のコメントを解除します。

```
$define DOC-FEATURES "TF"
```

5　手順 2 で削除したテキスト・インデックスを再作成します。たとえば、*i_blurbs* テキスト・インデックスを再作成する場合は、次のように入力します。

```
sp_create_text_index "KRAZYKAT", "i_blurbs", "blurbs", "", "copy"
```

ソート指定で使用するカラムの設定

style.vgw ファイルと *style.ufl* ファイルを修正してから、特定のカラムでソートします (ソート指定にカラムを含める方法については、「sort_by カラムを使用したソート順の指定」(44 ページ) を参照してください)。どちらのファイルも *$SYBASE/$SYBASE_FTS/collections/db.owner.index/style* にあります。ここで、*db.owner.index* は、データベース、データベース所有者、および sp_create_text_index を使用して作成したインデックスを表します。

たとえば、pubs2 データベースに i_blurbs というテキスト・インデックスを作成した場合、このファイルへのフル・パスは *$SYBASE/$SYBASE_FTS/collections/pubs2.dbo.i_blurbs/style* になります。

style.vgw ファイルと *style.ufl* ファイルを編集するには、次の手順に従います。

1 定義の追加先となるカラムを含んだテキスト・インデックスを削除します。テキスト・インデックスを削除しても、コレクション・ディレクトリは削除されません。

たとえば、blurbs テーブルの copy カラムに定義を追加するには、次のコマンドを使用してテキスト・インデックスを削除します。

```
sp_drop_text_index i_blurbs
```

2 *style.vgw* ファイルを編集します。次の行の下に移動します。

```
dda     "SybaseTextServer"
```

定義するカラムのエントリを追加します。構文は次のとおりです。ここで、*column_number* は、定義するカラムの番号です。

```
table: DOCUMENTS
{
    copy: fcolumn_number copy_column_number
}
```

カラムの番号は 0 から始まります。最初のカラムでソートする場合は、“f0”、2 番目のカラムでソートする場合は “f1”、3 番目のカラムでソートする場合は “f2” というように指定します。

たとえば、テーブルの最初のカラムを定義する場合、構文は次のようになります。

```
table: DOCUMENTS
{
          copy: f0 copy_f0
}
```

上記のように指定すると、*style.vgw* ファイルは次のようになります。

```
#
#       Sybase Text Server Gateway
#
$control: 1
gateway:
{
     dda:    "SybaseTextServer"
{
     copy: f0 copy_f0
}
}
```

3　*style.ufl* ファイルに、fts というデータ・テーブルのカラム定義を追加します。構文は次のとおりです。

```
data-table:        fts
{
     fixwidth:        copy_fcolumn_number precision datatype
}
```

カラムの番号は 0 から始まります。最初のカラムでソートする場合は、“f0”、2 番目のカラムでソートする場合は “f1”、3 番目のカラムでソートする場合は “f2” というように指定します。たとえば、精度 4、データ型 date という定義をテーブルの最初のカラムに追加するには、次のように入力します。

```
data-table: fts
{
        fixwidth:     copy_f0   4   date
}
```

同様に、精度 10、データ型 character という定義をテーブルの 2 番目のカラムに追加するには、次のように入力します。

```
data-table: fts
{
     fixwidth:     copy_f1   10   text
}
```

4　sp_create_text_index を使用して、インデックスを再作成します。

タグを含んだテキストへのフィルタの使用

タグを含んだドキュメント (HTML または PostScript など) に対して正確な検索を実行するには、テキスト・インデックスは、フィルタを使用してタグを取り除く必要があります。拡張型全文検索エンジンは、さまざまなドキュメント・タイプ (Microsoft Word、FrameMaker、WordPerfect、SGML、HTML など) 用のフィルタを備えています。

フィルタを使用するテキスト・インデックスを作成すると、ドキュメント内の、各タイプのタグのデータは、タグのタイプごとのドキュメント・ゾーンに置かれます。たとえば、“chapter” というタグがある場合は、その 1 つのドキュメント・ゾーンにすべての chapter 名が置かれます。ドキュメント全体を検索対象とするクエリと、“chapter” ゾーンのデータのみを検索するクエリのどちらも発行できます (詳細については、「in」(51 ページ) を参照してください)。

フィルタを使用するテキスト・インデックスを作成するには、そのテキスト・インデックスの *style.dft* ファイルを修正します。

1 sp_create_text_index を使用して、テキスト・インデックスを作成します。*option_string* パラメータには “empty” を使用します。これは、テキスト・インデックス用の *style.dft* ファイルは作成しても、Verity コレクションにデータを移植しないようにするためです。たとえば、blurbs テーブルの copy カラムのテキスト・インデックスを作成するには、次の構文を使用します。

```
sp_create_text_index "KRAZYKAT", "i_blurbs", "blurbs", "empty", "copy"
```

2 手順 1 で作成したテキスト・インデックスを削除します。これにより、テキスト・インデックスが削除されますが、*style.dft* ファイルは削除されません。たとえば、次のコマンドを使用して、i_blurbs テキスト・インデックスを削除します。

```
sp_drop_text_index i_blurbs
```

3 *style.dft* ファイルを編集します。*style.dft* ファイルは *$SYBASE/$SYBASE_FTS/collections/db.owner.index/style* ディレクトリにあります。*db.owner.index* は、データベース、データベース所有者、sp_create_text_index を使用して作成したインデックスを表します。たとえば、pubs2 データベースに i_blurbs というテキスト・インデックスを作成した場合、*style.dft* ファイルへのフル・パスは *$SYBASE/$SYBASE_FTS/collections/pubs2.dbo.i_blurbs/style* になります。

次の行の下に移動します。

```
field: f0
```

フィルタを使用するための構文を追加します。

- SGML ドキュメントの場合は、次のとおりです。

```
/filter="zone -nocharmap"
```

- HTML ドキュメントの場合は、次のとおりです。

```
/filter="zone -html -nocharmap"
```

すべてのドキュメント・タイプに次の構文を使用します。

```
/filter="universal"
```

たとえば、SGML ドキュメントの *style.dft* ファイルについては、次のようになります。

```
$control: 1
dft:
{
     field: f0
          /filter="zone -nocharmap"
     field: f1
     field: f2
     .
     .
     field: f15
{
```

SGML ドキュメントの *style.dft* ファイルについては、次のようになります。

```
$control: 1
dft:
{
     field: f0
          /filter="universal"
     field: f1
     field: f2
     .
     .
     field: f15
{
```

注意 データベースにドキュメント・データをロードするには、getsend を使用してください。getsend の引数は、database、table、column、row id です。挿入するテキストの各ローの rowid には null 値を挿入してください。フィルタを機能させるためには、getsend を使用して image カラムに挿入する必要があります。getsend の詳細については、*$SYBASE/$SYBASE_FTS/sample/source* ディレクトリの *README.TXT* ファイルと *getsend.c* ファイルを参照してください。

4　sp_create_text_index を使用して、インデックスを再作成します。次に例を示します。

```
sp_create_text_index "KRAZYKAT", "i_blurbs", "blurbs", "", "copy"
```

カスタム・シソーラスの作成

Verity では、thesaurus 演算子を使用すると、指定されたワードとその同義語を含むように検索が拡大されます (thesaurus 演算子の使用方法については、「thesaurus」(54 ページ) を参照してください)。アプリケーション固有の同義語が入ったカスタム・シソーラスを作成して、デフォルトのシソーラスの代わりに使用できます。

たとえば、デフォルトのシソーラス (英語版) には、“money” の同義語として “cash”、“currency”、“lucre”、“wampum”、“greenbacks” というワードが入っています。さらに、“money” の同義語として別のワード、たとえば “bid”、“tokens”、“credit”、“asset”、“verbal offer” が入ったカスタム・シソーラスを作成できます。

カスタム・シソーラスを作成するには、次の手順に従います。

1 アプリケーションで使用する同義語のリストを作成します。デフォルトのシソーラスを確認すると役に立つ場合もあります (「デフォルトのシソーラスを確認する (オプション)」(33 ページ) を参照してください)。

2 制御ファイルを作成して、カスタム・シソーラス用に定義する同義語を登録します (「制御ファイルの作成」(33 ページ) を参照してください)。

3 mksyd ユーティリティを使用して、カスタム・シソーラスを作成します (「シソーラスの作成」(34 ページ) を参照してください)。mksyd ユーティリティは *$SYBASE/$SYBASE_FTS/verity/<verity_platform_directory>/bin* にあります。

   制御ファイルを入力として使用します。

4 デフォルトのシソーラスを、作成したカスタム・シソーラスに置き換えます (「デフォルトのシソーラスをカスタム・シソーラスに置き換える」(35 ページ) を参照してください)。

「カスタム・シソーラスのサポート」と mksyd ユーティリティの詳細については、Verity Web サイト (http://www.verity.com) を参照してください。

次の 2 つのサンプル・ファイルは、カスタム・シソーラスの設定方法と使用方法を示しています。

- *sample_text_thesaurus.ctl* は、制御ファイルのサンプルです。
- *sample_text_thesaurus.sql* は、制御ファイルのサンプルで定義されたカスタム・シソーラスに対してクエリを発行します。

上記のファイルは、*$SYBASE/$SYBASE_FTS/sample/scripts* ディレクトリにあります。

デフォルトのシソーラスを確認する (オプション)

制御ファイルには、シソーラスの同義語の定義がすべて入っています。デフォルトのシソーラスを確認するには、mksyd ユーティリティを使用してシソーラスの制御ファイルを作成します。mksyd ユーティリティは *$SYBASE/SYBASE_FTS/verity/<verity_platform_directory>/bin* にあります。

次の構文を使用します。

```
mksyd -dump -syd
$SYBASE/$SYBASE_FTS/verity/common/vdkLanguage/vdk20.syd
-f work_location/control_file.ctl
```

各パラメータの意味は、次のとおりです。

- *vdkLanguage* － vdkLanguage 設定パラメータの値 (たとえば、“english”)
- *work_location* － 制御ファイルを配置するディレクトリ
- *control_file* － デフォルトのシソーラスから作成する制御ファイルの名前

デフォルトの同義語リストを表示するために、作成した制御ファイル (*control_file.ctl*) を確認します。

制御ファイルの作成

カスタム・シソーラス用の新しい同義語が入った制御ファイルを作成します。制御ファイルは、構造化フォーマットの ASCII テキスト・ファイルです。テキスト・エディタ (vi や emacs など) を使用して、次のいずれかの操作を行います。

- デフォルトのシソーラスの制御ファイルを編集して、既存のシソーラスに新しい同義語を追加する (「デフォルトのシソーラスを確認する (オプション)」 (33 ページ) を参照してください)。
- 独自の同義語のみが入った新しい制御ファイルを作成する。

制御ファイルの構文

制御ファイルの synonyms: 文に、同義語リストの定義があります。たとえば、次に示すのは *colors.ctl* という名前の制御ファイルです。

```
$control: 1
synonyms:
{
list: "red, ruby, scarlet, fuchsia,¥
magenta"
list: "electric blue <or> azure"
/keys = "lapis"
}
$$
```

synonyms: 文には、次のものが含まれます。

- list: キーワード。同義語リストの開始を指定します。リストに記述する同義語は、クエリの形式か、カンマで区切られたワードまたはフレーズのリストの形式です。
- 各 list: には、オプションで、カンマで区切られた 1 つ以上のキーを指定する /keys 変更子を付けることができます。上記の例では、最初の “list” にキーは指定されていません。つまり、“red”、“ruby”、“scarlet”、“fuchsia”、または “magenta” というワードでシソーラスに問い合わせると、リストが検出されます。2 番目の “list” は /keys 変更子を使用して、キーを 1 つ指定しています。したがって、<thesaurus>lapis を指定した場合にのみ、リスト内のワードまたはフレーズがクエリの条件を満たします。

注意 emacs を使用して同義語リストを作成したときに、リストが 1 行より長くなる場合は、auto-fill モードをオフにしてください。リストが複数行に分かれる場合は、各行の最後に円記号 (¥) を付けて、複数の行を 1 つのリストとして処理できるようにします。

さらに複雑な制御ファイルの例については、Verity Web サイトを参照してください。

シソーラスの作成

mksyd ユーティリティでは、制御ファイルを入力としてカスタム・シソーラスを作成します。このユーティリティは、次のロケーションにあります。

$SYBASE/$SYBASE_FTS/verity/<verity_platform_directory>/bin

この *bin* ディレクトリから、mksyd を実行するか、実行するエイリアスを定義します。任意のワーク・ディレクトリにカスタム・シソーラスを作成できます。

カスタム・シソーラスを作成するための mksyd の構文は、次のとおりです。

```
mksyd -f control_file.ctl -syd custom_thesaurus.syd
```

各パラメータの意味は、次のとおりです。

- *control_file* − 前の項で作成した制御ファイルの名前
- *custom_thesaurus* − 作成するカスタム・シソーラスの名前

たとえば、上記で定義した制御ファイルのサンプルを読み込んで mksyd ユーティリティを実行し、ワーク・ディレクトリに出力するには、次の構文を使用します。

```
mksyd -f /usr/u/sybase/dba/thesaurus/colors.ctl
-syd /usr/u/sybase/dba/thesaurus/custom.syd
```

デフォルトのシソーラスをカスタム・シソーラスに置き換える

vdk20.syd という名前のデフォルトのシソーラスが *$SYBASE/$SYBASE_FTS/verity/common/vdkLanguage* にあります。*vdkLanguage* は、vdkLanguage 設定パラメータの値です (たとえば、英語のディレクトリは *$SYBASE/$SYBASE_FTS/verity/common/english* です)。アプリケーションおよびユーザはそれぞれ実行時に、このロケーションからデフォルトのシソーラスを読み込んで使用します。デフォルトのシソーラスをカスタム・シソーラスと置き換えるには、次の手順に従います。

1　デフォルトのシソーラスをバックアップしてから、カスタム・シソーラスと置き換えます。次に例を示します。

```
mv /$SYBASE/$SYBASE_FTS/verity/common/english/vdk20.syd default.syd
```

2　*vdk20.syd* ファイルをカスタム・シソーラスと置き換えます。次に例を示します。

```
cp custom.syd /$SYBASE/$SYBASE_FTS/verity/common/english/vdk20.syd
```

3　制御ファイル (*control_file.ctl*) の検査を再起動します。設定ファイルへの変更は必要ありません。クエリの実行時ではなく、制御ファイル (*control_file.ctl*) の検査の起動時に、このロケーションからシソーラスが読み込まれます。

これで、thesaurus 演算子を使用するクエリで、カスタム・シソーラスを使用できます。

トピックの作成

この項では、Verity のトピックについて概要を簡単に説明します。トピックの詳細については、「第 8 章 Verity のトピック」を参照してください。

トピックとは、概念またはサブジェクト領域に関連する情報をグループ化したものです。適切なトピック定義があれば、そのトピックを検索すればよいので、ユーザは複雑な構文を使用してクエリを作成する必要がありません。

ユーザは、ワードとフレーズ、Verity の演算子と変更子、ウェイト値を組み合わせてトピックを作成します。こうしてトピックを作成すると、どのユーザでもトピックへの問い合わせができます。

アプリケーションの要求を特定し、命名規則、および以下のファイルやディレクトリのロケーションに関する基準を確立してから、トピックを作成します。

- アウトライン・ファイル ― トピック定義が含まれるファイル。トピックごとにアウトライン・ファイルを持ちます。
- トピック・セット・ディレクトリ ― コンパイルしたトピックが格納されるディレクトリ。トピックごとにトピック・セット・ディレクトリを持ちます。
- 知識ベース・マップ・ファイル ― トピック・セット・ディレクトリへのポインタが格納されるファイル。

トピックを実装するには、次の手順を実行します。

1 トピックを定義するためのアウトライン入力ファイルを 1 つ以上作成します「アウトライン・ファイルの作成」(37 ページ) を参照してください。トピック・セットを 1 つ移植するごとに、アウトライン・ファイルが 1 つ使用されます。

2 mktopics ユーティリティを使用して、トピック・セット・ディレクトリの作成と移植を行います (「トピック・セット・ディレクトリの作成」(38 ページ) を参照してください)。mktopics ユーティリティは *$SYBASE/$SYBASE_FTS/verity/<verity_platform_directory>/bin* にあります。

   各トピック・セット・ディレクトリは、トピックのアウトライン入力ファイルごとに移植されます。

3 知識ベース・マップを作成して、トピック・セット・ディレクトリのロケーションを 1 つ以上指定します (「知識ベース・マップの作成」(38 ページ) を参照してください)。

4 knowledge_base 設定パラメータを設定して、知識ベース・マップのロケーションを指定します (「知識ベース・マップのロケーションの定義」(39 ページ) を参照してください)。

5 定義されたトピックに対してクエリを実行します。

トピックの機能は、以下のサンプル・ファイルに示されています。

- *sample_text_topics.otl* は、アウトライン・ファイルのサンプルです。
- *sample_text_topics.kbm* は、知識ベース・マップのサンプルです。
- *sample_text_topics.sql* は、定義されたトピックを使用してクエリを発行します。

上記のファイルは、*$SYBASE/$SYBASE_FTS/sample/scripts* ディレクトリにあります。

アウトライン・ファイルの作成

トピックのアウトライン・ファイルでは、トピックを使用してクエリを発行するときに、検索エンジンが使用するワードとフレーズ、Verity の演算子と変更子、ウェイト値の組み合わせをすべて指定します。アウトライン・ファイルは、構造化フォーマットの ASCII テキスト・ファイルです。

たとえば、次のアウトライン・ファイルは、“saint-bernard” というトピックを定義します。

```
$control: 1
saint-bernard <accrue>
*0.80 "Saint Bernard"
*0.80 "St. Bernard"
* "working dogs"
* "large dogs"
* "European breeds"
$$
```

トピック “saint-bernard” を指定したクエリを発行すると、拡張型全文検索エンジンは次の処理を行います。

- “Saint Bernard”、“St. Bernard”、“working dogs”、“large dogs”、“European breeds” というフレーズを 1 つ以上含んだドキュメントを返す
- “Saint Bernard” または “St. Bernard” というフレーズを含んだドキュメントに、“working dogs”、“large dogs”、または “European breeds” というフレーズを含んだドキュメントよりも高いスコアを付ける

この例は、非常に基本的なトピック定義です。アウトライン・ファイルでさらに複雑な関係を作成する場合は、以下を使用します。

- 複数レベルのサブトピック
- Verity 演算子の組み合わせ (この例では accrue を使用)
- Verity 変更子

注意　Windows NT では、Verity から提供されている Verity Intelligent Classifier 製品の GUI (グラフィカル・ユーザ・インタフェース) を使用して、トピックのアウトラインを作成できます。Intelligent Classifier を使用すると、トピック・セット・ディレクトリが自動的に作成されるので、「知識ベース・マップの作成」(38 ページ) でのトピックの設定から始められます。

トピック・セット・ディレクトリの作成

mktopics ユーティリティを使用して、トピック・セット・ディレクトリの作成と移植を行います。このユーティリティは *$SYBASE/$SYBASE_FTS/verity/<verity_platform_directory>/bin* にあります。

この *bin* ディレクトリから、mktopics を実行するか、実行するエイリアスを定義します。任意のワーク・ディレクトリにトピック・セット・ディレクトリを作成できます。

mktopics の構文は、次のとおりです。

mktopics -outline *outline_file*.otl -topicset *topic_set_directory*

各パラメータの意味は、次のとおりです。

- *outline_file* – 「アウトライン・ファイルの作成」(37 ページ) で作成したアウトライン・ファイルの名前
- *topic_set_directory* – 作成するトピック・セット・ディレクトリの名前

たとえば、前述の *saint-bernard.otl* ファイルを読み込んで mktopics ユーティリティを実行し、ワーク・ディレクトリに出力するには、次の構文を使用します。

```
mktopics -outline
/usr/u/sybase/topic_outlines/saint-bernard.otl
-topicset /usr/u/sybase/topic_sets/
saint-bernard_topic
```

知識ベース・マップの作成

知識ベース・マップでは、1 つ以上のトピック・セット・ディレクトリのロケーションを指定します。トピック・セットへの完全に修飾されたディレクトリ・パスを定義する ASCII 知識ベース・マップ・ファイルを作成します。

たとえば、次の知識ベース・マップ・ファイルは、マップ内の複数の知識ベースをリストする方法を示しています。最初のエントリは、前述の mktopics で作成されたトピック・セット・ディレクトリを示します。

```
$control:1
kbases:
{
kb:
/kb-path = /usr/u/sybase/topic_sets/saint-bernard_topic
kb:
/kb-path = /usr/u/sybase/topic_sets/another_topic
}
```

知識ベース・マップのロケーションの定義

knowledge_base 設定パラメータを設定して、知識ベース・マップのロケーションを指定します。次に例を示します。

```
sp_text_configure KRAZYKAT, 'knowledge_base',
'/usr/u/sybase/topic_sets/sample_text_topics.kbm'
```

knowledge_base 設定パラメータは静的なパラメータなので、定義を有効にするには拡張型全文検索エンジンを再起動してください。

定義済みトピックに対するクエリの実行

定義済みのトピックを使用してクエリを実行できるので、複雑なクエリを実行する必要はありません。たとえば、“saint-bernard” というトピックを作成する前は、次の構文を使用する必要があります。

```
...where i.index_any = "<accrue> ([80]Saint Bernard, [80]St. Bernard, working
dogs, large dogs, European breeds)"
```

その結果、次のようなドキュメントが見つかります。

- “Saint Bernard”、“St. Bernard”、“working dogs”、“large dogs”、“European breeds” というフレーズを 1 つ以上含んだドキュメント
- “Saint Bernard” または “St. Bernard” というフレーズを含んだドキュメントに、“working dogs”、“large dogs”、または “European breeds” というフレーズを含んだドキュメントよりも高いスコアを付けたドキュメント

トピック “saint-bernard” を作成した後は、次の構文を使用できます。

```
...where i.index_any = "<topic>saint-bernard"
```

または

```
...where i.index_any = "saint bernard"
```

注意　クエリ式の中にワードを入力した場合は、拡張型全文検索エンジンではそのワードをトピックの名前と一致させようとします。クエリ式の中にフレーズを入力した場合は、拡張型全文検索エンジンではスペースをハイフン (-) に置き換えてから、そのフレーズをトピックの名前と一致させようとします。たとえば、拡張型全文検索エンジンは、“saint bernard” をトピック “saint-bernard” と一致させます。

クエリでトピックを使用する例については、*sample_text_topics.sql* ファイルを参照してください。

トピックのトラブルシューティング

knowledge_base 設定パラメータによって、存在しない知識ベース・マップ・ファイルを指定した場合は、拡張型全文検索エンジンが Verity とのセッションを開始できないため、サーバは起動しません。マップ・ファイルが存在しても無効なエントリがある場合は、Verity は起動時に警告メッセージを発行します。エラーの訂正は、*$SYBASE* ディレクトリの *<textserver>.cfg* ファイルを編集することによってできます。パス情報の訂正と、“knowledge_base=” で始まる行の変更ができます。

第 5 章

全文検索クエリの作成

この章では、全文検索で使用できる疑似カラム、検索演算子、変更子について説明します。

全文検索クエリのコンポーネント

全文検索クエリを作成するには、Adaptive Server の select 文の一部として検索パラメータを入力します。このパラメータによって、拡張型全文検索エンジンは検索を処理します。select 文には、次のものが必要です。

- Verity 言語クエリを index_any 疑似カラムに割り当てる where 句
- 検索パラメータをさらに詳細に定義する疑似カラム (オプション)
- 送信元テーブルの IDENTITY カラムまたはプライマリ・キーとインデックス・テーブルの id カラムのジョイン

たとえば、blurbs テーブルの copy カラムに、“software” というワードが最も現れる上位 10 個のドキュメントを返すには、次のように入力します。

```
select t1.score, t2.copy
from i_blurbs t1, blurbs t2
where t1.id=t2.id
and t1.index_any = "<many> <word> software"
and t1.max_docs = 10
```

Adaptive Server は、検索を処理する拡張型全文検索エンジンに Verity クエリを渡します。詳細については、「全文検索の動作の仕組み」(8 ページ)を参照してください。

デフォルトの動作

デフォルトの、または単純な、拡張型全文検索エンジンへのクエリの構文は、大まかには次のように処理されます。

1 検索は、大文字と小文字を区別する

2 STEM 演算子が検索ワードに適用される

3 MANY 変更子が適用される

4 ACCRUE 演算子が、親レベルでアクティブになる

インデックス・テーブルの疑似カラム

疑似カラムは、検索のパラメータを定義し、結果データへのアクセスを提供するインデックス・テーブルのカラムです。詳細については、「インデックス・テーブル」(5 ページ) を参照してください。このカラムは、クエリのコンテキストでのみ有効です。つまり、カラム内の情報はクエリの間のみ有効です。後続のクエリが異なるパラメータ・セットを持つ場合は、疑似カラムには異なる値セットが入ります。

インデックス・テーブルの各疑似カラムは、それぞれ異なる検索属性を表します。たとえば、score カラムを指定する場合、クエリは定義されたパラメータの範囲内の結果セットのみを表示します。たとえば次のクエリは、90 より大きい score 値を持つ結果のみを表示します。

```
index_table_name.score > 90
```

特定のドキュメントについて Verity によって生成されたデータを検索するために、他の疑似カラム (highlight など) が使用されます。表 5-1 は、拡張型全文検索エンジンによって管理されている疑似カラムの説明です。

表 *5-1:* 拡張型全文検索エンジンの疑似カラム

疑似カラム名	説明	データ型	長さ (バイト数)
cluster_number	そのローが一部分を構成するクラスタを示す。クラスタには 1 から順に番号が付けられます。cluster_number column カラムは、クエリの select 句でのみ使用できる。	int	4
cluster_keywords	Verity がクラスタの作成に使用するキーワードを示す。cluster_keywords は、クエリの select 句でのみ使用できる。	varchar	255
highlight	ドキュメント内でクエリからのすべてのワードを補う。highlight は、クエリの select 句でのみ使用できる。	text	16
id	コレクション内のドキュメントをユニークに識別する。送信元テーブルの IDENTITY カラムとのジョインに使用される。id は、クエリの select 句または where 句で使用できる。	numeric	6

疑似カラム名	説明	データ型	長さ (バイト数)
index_any	拡張型全文検索エンジンに、Verity 言語クエリを与える。index_any は、where 句でのみ使用できる。擬似カラムは char(255) で定義されていますが、index_any 句の最大長は 16000 です。	varchar	255
max_docs	デフォルトのソート順に基づいて、結果を最初から *n* ドキュメントに制限する。クラスタード結果セットでは、クラスタごとに、結果を最初から *n* ドキュメントに制限する。max_docs は、where 句でのみ使用できる。	int	4
score	検索文字列とインデックス・カラムとの間の相関の標準尺度。特定のドキュメントに関連する score は、そのドキュメントの検索に使用されたクエリに関してのみ意味を持つ。score は、select 句または where 句で使用できる。	int	4
sort_by	結果セットを返すときのソート順を指定する。 拡張型全文検索エンジンでは、sort_by カラムで 16 個までソート指定を設定できる。 sort_by は、where 句でのみ使用できる。	varchar	35
summary	要約データを選択する。summary カラムは、クエリの select 句でのみ使用できる。	varchar	255
total_docs	検索基準と一致したドキュメントの総数が含まれる。	int	4

次の項では、疑似カラムの機能について説明します。

score カラムを使用した、検索結果の関連性によるランク付け

関連性によるランク付けは、ドキュメントがクエリを満たす程度を示す値を score パラメータに割り当てる、拡張型全文検索エンジンの機能です。score の計算は、クエリで使用される検索演算子によって異なります。詳細については、「Verity 演算子の使用」(50 ページ) を参照してください。ドキュメントがクエリをより満たすほど、ドキュメントに対する score 値は高くなります。

たとえば、“rain” というワードを含むドキュメントを検索する場合、“rain” を 12 個含むドキュメントは、“rain” を 6 個含むドキュメントよりも高い score 値になります。

クエリで *score* を高い値 (90 など) に設定すると、結果セットはその数字よりも大きい *score* 値を持つドキュメントに制限されます。

注意　score 値の表現には、Verity は小数を使用しますが、Sybase は整数を使用します。たとえば、Verity が score の値として .85 をレポートする場合、Sybase は同じ値を 85 とレポートします。

たとえば次のクエリは、“raconteur” か “Paris” というワード、またはその両方を含み、score が 90 以上のドキュメントを検索します。

```
select t1.score, t2.copy
from i_blurbs t1, blurbs t2
where t1.id=t2.id and t1.score > 90
and t1.index_any = "<accrue>(raconteur, Paris)"
score     copy
-----     ------------------------------------------------------------
(0 rows affected)
```

このクエリは、“raconteur” というワードまたは “Paris” というワードを含み、かつ score が 90 より大きいドキュメントを見つけられません。しかし、クエリの *score* 値を 39 に下げると、*blurbs* テーブルの 1 つのドキュメントが “aconteur” または “Paris” というワードを含んでいることがわかります。

```
select t1.score, t2.copy
from i_blurbs t1, blurbs t2
where t1.id=t2.id and t1.score > 39
and t1.index_any = "<accrue>(raconteur, Paris)"
score     copy
-----     ------------------------------------------------------------
40        A chef's chef and a raconteur's raconteur, Reginald
          Blotchet-Halls calls London his second home. "Th' palace
    . . .
```

sort_by カラムを使用したソート順の指定

ソート順は、結果セット内のデータを順序付けるために使用する照合順を指定します。デフォルトのソート順は、sort_order 設定パラメータで設定されます。詳細については、「デフォルトのソート順の設定」(67 ページ) を参照してください。大文字と小文字を区別しないソート順がサポートされています。

sort_by 疑似カラムを使用すると、デフォルトとは異なるソート順で結果セットを返します。sort_by 疑似カラムでは 16 個までソート指定を指定できます。

表 5-2 に、sort_by 疑似カラムの値を示します。

表 *5-2: sort_by* 疑似カラムの値

値	戻り値
fts_score desc	score で降順にソートした結果セット。
fts_score asc	score で昇順にソートした結果セット。
fts_timestamp desc	タイムスタンプで降順にソートした結果セット。
fts_timestamp asc	タイムスタンプで昇順にソートした結果セット。
column_name desc	カラムで降順にソートした結果セット。*column_name* は送信元テーブルのカラムの名前。
column_name asc	カラムで昇順にソートした結果セット。*column_name* は送信元テーブルのカラムの名前。
fts_cluster asc	クラスタード結果セット。詳細については、「疑似カラムを使用した、クラスタード結果セットの要求」(46 ページ) を参照。

注意　*style.vgw* ファイルと *style.ufl* ファイルを修正してから、特定のカラムでソートします。詳細については、「ソート指定で使用するカラムの設定」(28 ページ) を参照してください。

たとえば、次のクエリはドキュメントをタイムスタンプで昇順にソートします。

```
select t1.score, t2.copy
from i_blurbs t1, blurbs t2
where t1.id=t2.id and t1.score > 90
and t1.index_any = "<accrue>(raconteur, Paris)"
and t1.sort_by = "fts_timestamp asc"
```

summary カラムを使用したドキュメントの要約

summary 疑似カラムを使用すると、クエリは、検索基準を満たすドキュメントの全体ではなく、要約のみを返します。summary カラムはデフォルトでは使用できません。*style.prm* ファイルを編集してから、テキスト・インデックスを作成して要約を有効にします。詳細については、「QBE (例示による問い合わせ)、要約、クラスタリングの有効化」(25 ページ) を参照してください。

たとえば次のクエリは、“Iranian” というワードと “book” というワードを含むドキュメントの要約のみを返します (この例では、*style.prm* ファイルは 255 文字表示するように設定されています)。

```
select t1.score, t1.summary
from i_blurbs t1, blurbs t2
where t1.id=t2.id and t1.score > 70
and t1.index_any = "(Iranian <and> book)"

score       summary
-----       ------------------------------------------------------------
78          They asked me to write about myself and my book, so here goes:
            I started a restaurant called "de Gustibus" with two of my fri

(1 row affected)
```

拡張型全文検索エンジンは、255 バイトまでの要約をサポートします。

要約を使用するクエリの別の例については、*$SYBASE/$SYBASE_FTS/sample/scripts* ディレクトリのサンプル・スクリプト *sample_text_queries.sql* を参照してください。

疑似カラムを使用した、クラスタード結果セットの要求

クラスタリング機能は、結果セットを分析し、ローをクラスタにグループ化します。その結果、各クラスタ内のローは、平均して、他のクラスタ内のローよりもセマンティック上互いに似ています。ローをサブトピックで順序付けすると、結果セットに含まれる主要なサブジェクト領域がわかりやすくなります。

クラスタード結果セットを返すと、ノンクラスタード結果セットを返すよりも非常に遅くなる場合があります。クエリの応答時間が重要な場合には、ノンクラスタード結果セットを使用します。

クラスタリング使用の準備

クラスタリング機能を有効にしてテキスト・インデックスを作成してから、クラスタード結果セットを要求します。詳細については、「QBE (例示による問い合わせ)、要約、クラスタリングの有効化」(25 ページ) を参照してください。

Verity のクラスタリングのアルゴリズムは、次の設定パラメータの値に基づいて、類似したローごとにグループ化します。

- cluster_style
- cluster_max
- cluster_effort
- cluster_order

sp_text_cluster を使用すると、これらの設定パラメータについて、デフォルト値とは異なる値をクエリで使用できます。詳細については、sp_text_cluster (141 ページ) を参照してください。

クラスタード結果セットを要求するクエリの作成

クラスタード結果セットを取得するには、クエリの sort_by 疑似カラムに、ソート指定として “fts_cluster” を指定します。次に例を示します。

```
select t1.score, t2.copy
from i_blurbs t1, blurbs t2
where t1.id=t2.id
and t1.index_any = "<many> <word> software"
and t1.max_docs = 10
and t1.sort_by = "fts_cluster asc"
```

次の疑似カラムをクエリで使用すると、追加のクラスタリング情報を返します。

- cluster_number － ローが属するクラスタの番号を格納します。クラスタには 1 から順に番号が付けられます。
- cluster_keywords － クラスタ内で最も多く見つかるワードを格納します。どのクラスタにも属していないローに対しては、cluster_keywords カラムに null 値が入ります。

- max_docs − 各クラスタに返されるローの数を制限します。ノンクラスタード・クエリでは、max_docs カラムは、結果セットに返されるロー数の合計を制限します。
- score − 0 ～ 10000 までの値を格納します。score の値が高いほど、ローはクラスタの中心に近づきます。score 値 0 は、ローがどのクラスタにも属していないことを示します。ノンクラスタード・クエリでは、score カラムには 0 ～ 100 の値が入ります。検索エンジンは、score 値 0 の場合は結果を返しません。score 値 0 は「一致なし」を表しますが、ユーザに score 値 0 が示されることはありません。

クラスタリングを使用する SQL 文の例については、*$SYBASE/$SYBASE_FTS/sample/scripts* ディレクトリのサンプル・スクリプト *sample_text_queries.sql* を参照してください。

全文検索演算子

全文検索を実行するために使用する特別な検索演算子は、Verity 検索エンジンの一部です。表 5-3 は、拡張型全文検索エンジンが提供する Verity 検索演算子を示します。

注意　Sybase では、フィールドへの XML 要素のインデックス付けはサポートしていません。

表 *5-3: Verity* 検索演算子

演算子名	説明
accrue	クエリで指定した検索要素が少なくとも 1 つ含まれるドキュメントを選択する。検索要素が多いほどスコアは高くなる。
and	クエリで指定した検索要素がすべて含まれるドキュメントを選択する。
complement	score 値の補数を返す (100 から score 値を減算)。
in	指定したドキュメント・ゾーン内に検索基準が含まれるドキュメントを選択する。
like	クエリで指定したサンプル・ドキュメントまたはパッセージに似ているドキュメントを選択する。
near	指定した検索要素が含まれるドキュメントを選択する。検索ワード同士がドキュメント内で近くにあるほど、ドキュメントのスコアは高くなる。
near/n	互いに *n* ワード以内の範囲に 2 つ以上の検索ワードが含まれるドキュメントを選択する。*n* は 1000 までの整数。検索ワード同士がドキュメント内で近くにあるほど、ドキュメントのスコアは高くなる。
or	クエリで指定した検索要素が少なくとも 1 つ含まれるドキュメントを選択する。
paragraph	指定したすべての検索要素が同じ段落に含まれるドキュメントを選択する。
phrase	特定のフレーズが含まれるドキュメントを選択する。フレーズは、特定の順序で並ぶ 2 つ以上のワードのグループです。
product	検索基準の項目それぞれの score 値を乗算する。
sentence	指定したすべてのワードが同じ文に含まれるドキュメントを選択する。

演算子名	説明
stem	指定したワードとその語尾変化したものが含まれるように検索を拡大する。
sum	検索基準のすべての項目に score 値を追加する。
thesaurus	指定したワードとその同義語が含まれるように検索を拡大する。
topic	入力した検索ワードがトピックであることを指定する。
typo/n	指定したワードとそれに類似したワードが含まれるように検索を拡大する。オプションの *n* 変数では、検索対象ワードと突き合わせたワードの間の最大誤り数を指定する。
wildcard	検索文字列に含まれるワイルドカード文字を一致させる。特定の文字は自動的にワイルドカード指定を示す。
word	基本的なワード検索を実行し、指定したワードのインスタンスが 1 つ以上含まれるドキュメントを選択する。
yesno	0 以外の score 値をすべて 100 に変換する。

Verity 演算子を使用するときの考慮事項

全文検索クエリを作成するときは、次の点を考慮します。

- Verity では、ダッシュ、アンダースコア、およびアンパサンドはワード・セパレータではなくアルファベット文字として扱われます。ロケール固有の定義ファイルを変更することで、この動作を変更できます。たとえば、-tokenized_as_alphabet-& は、-tokenized_as_alphabet_& に変更できます。
- クエリ内の演算子は山カッコ (<>) で囲みます。山カッコで囲まない場合、拡張型全文検索エンジンは次のようなエラー・メッセージを発行します。

```
Msg 20200, Level 15, State 0:
Server 'KRAZYKAT', Line 1:
Error   E1-0111 (Query Builder): Syntax error in query string near character 5
Msg 20200, Level 15, State 0:
Server 'KRAZYKAT', Line 1:
Error   E1-0114 (Query Builder): Error parsing query: word(tasmanian)
Msg 20101, Level 15, State 0:
Server 'KRAZYKAT', Line 1:
VdkSearchNew failed with vdk error (-40).
Msg 20101, Level 15, State 0:
Server 'KRAZYKAT', Line 1:
VdkSearchGetInfo failed with vdk error (-11).
score     copy
-----     ------------------------------------------------------------
(0 rows affected) score
```

- Verity 言語クエリは、一重引用符 (') または二重引用符 (") で囲む必要があります。拡張型全文検索エンジンは、最も外側にある引用符を取り除いてから、クエリを Verity に送信します。たとえば、次のようなクエリを入力します。

  ```
  ...where index_any = "'?own'"
  ```

  拡張型全文検索エンジンは、次のクエリを Verity に送信します。

  ```
  '?own'
  ```

- SQL で、クエリが and でつながれた複数の “index_any” 句で構成される場合があります。適切な and 値の文字列では、プレフィクスとして “<snnn>” を付けることができます。拡張型全文検索において、それらの文字列は “nnn” の値の順にすべて連結されます。“<snnn>” は削除されます。たとえば、次のようなクエリがあります。

```
where index_any="<s001>hello"
and index_any="<s002> world"
```

これは、次のクエリと同じです。

```
where index_any = "hello world"
```

これにより、255 文字を超える検索文字列を避けることができます。

- 大文字と小文字が混在する検索ワードは、自動的に大文字と小文字が区別されます。すべて大文字またはすべて小文字で入力された検索ワードは、自動的には大文字と小文字が区別されません。たとえば、“Server” というクエリでは、文字列 “Server” のみが検索されます。“server” または “SERVER” というクエリでは、文字列 “Server”、“server”、“SERVER” が検索されます。
- 表 5-4 に示すように、クエリ式に代替構文を使用することもできます。

表 *5-4: Verity* の代替構文

通常のクエリ式	代替構文
<MANY><WORD>*string*	"*string*"
<MANY><STEM>*string*	'*string*'

代替構文を使用する場合、拡張型全文検索エンジンは最も外側にある引用符を取り除いてから、クエリを Verity に送信する点に注意してください。たとえば、次のようなクエリを入力します。

```
...where index_any = "'play'"
```

拡張型全文検索エンジンは、次のクエリを Verity に送信します。

```
'play'
```

これは、次のクエリと同じです。

```
<MANY><STEM>play
```

Verity 演算子の使用

次の項では、表 5-3 (47 ページ) で示した Verity 演算子の使用方法について説明します。

accrue

accrue 演算子は、クエリで指定した検索項目が少なくとも 1 つ含まれるドキュメントを選択します。検索要素は 2 つ以上必要です。各結果は、関連性によってランク付けされます。たとえば、次のクエリは、blurbs テーブルの copy カラムで “restaurant” または “deli”、あるいはその両方のワードを検索します。

```
select t1.score, t2.copy
from i_blurbs t1, blurbs t2
where t1.id=t2.id and t1.score > 35
and t1.index_any = "<accrue>(restaurant, deli)"
```

and、*or*

and 演算子および or 演算子は、指定した検索要素が含まれるドキュメントを選択します。各結果は、関連性によってランク付けされます。and 演算子は、クエリで指定した要素がすべて含まれるドキュメントを選択します。たとえば、次のクエリは、“Iranian” と “business” の両方が含まれるドキュメントを選択します。

```
select t2.copy
from i_blurbs t1, blurbs t2
where t1.id=t2.id
and t1.index_any = "(Iranian <and> business)"
```

or 演算子は、検索要素のいずれかが含まれるドキュメントを選択します。たとえば、前述のクエリを or 演算子を使用して書き換えると、“Iranian” または “business” というワードが含まれるドキュメントが選択されます。

```
select t2.copy
from i_blurbs t1, blurbs t2
where t1.id=t2.id
and t1.index_any = "(Iranian <or> business)"
```

complement

complement 演算子は、ドキュメントの score 値の補数を返します。つまり、score の値を 100 から引いて、結果をそのドキュメントの score 値として返します。

in

in 演算子は、1 つ以上のドキュメント・ゾーンに指定した検索要素が含まれるドキュメントを選択します。ドキュメント・ゾーンは、テキスト・インデックスに対して次の 2 つのシナリオで作成されます。

- sp_create_text_index を使用して複数のカラムにインデックスを作成する場合、各カラムについてテキスト・インデックス内にドキュメント・ゾーンが作成されます。詳細については、「テキスト・インデックス作成時の複数カラムの指定」(20 ページ) を参照してください。1 つのカラムにインデックスを作成する場合は、ドキュメント・ゾーンは作成されません。たとえば、テキスト・インデックスを作成するときに blurbs テーブルの au_id カラムと copy カラムを指定する場合、次のクエリを発行できます。

  ```
  select t1.score, t2.copy
  from i_blurbs t1, blurbs t2
  where t1.id=t2.id and t1.score > 35
  and t1.index_any = "gorilla <in> copy"
  ```

  これにより、copy カラムにワード “gorilla” が含まれるローが返されます。しかし、テキスト・インデックスを作成するときに blurbs テーブルの copy カラムのみを指定した場合、このクエリではローは何も返されません。

- フィルタを使用するインデックスを作成するとき、ドキュメント内の各タグについてドキュメント・ゾーンが作成されます。詳細については、「タグを含んだテキストへのフィルタの使用」(30 ページ) を参照してください。in 演算子の後ろにタグ名を指定することによって、検索を特定のタグに制限することができます。たとえば、HTML ドキュメント内の “title” タグでワード “automotive” を検索するには、次のように指定します。

  ```
  select t1.score, t2.copy
  from i_blurbs t1, blurbs t2
  where t1.id=t2.id and t1.score > 35
  and t1.index_any = "automotive <in> title"
  ```

フィルタを使用するテキスト・インデックスに指定できるカラムは 1 つのみです。

like

like 演算子は、指定したドキュメントまたはパッセージに似ているドキュメントを選択します。検索エンジンはテキストを分析し、最も重要な用語を見つけて使用します。複数のサンプルを指定した場合、検索エンジンはサンプル間で共通する重要な用語を選択します。各結果は、関連性によってランク付けされます。

like 演算子は、オペランドを 1 つ使用します。これは、QBE (例示による問い合わせ) フォームと呼ばれます。QBE フォームは、リテラル・テキストまたはドキュメント ID のいずれかです。ドキュメント ID は、送信元テーブルの IDENTITY カラムから設定されます。たとえば、IDENTITY が “2” のローの copy カラムのドキュメントに似ているドキュメントを選択するには、次のように入力します。

```
select t1.score, t2.copy
from i_blurbs t1, blurbs t2
where t1.id=t2.id and t1.score > 35
and t1.index_any = '<like> ( "{2}" )'
```

style.prm ファイルの次の行のコメントを解除してから、QBE フォームでリテラル・テキストを使用します。

```
$define DOC-FEATURES "TF"
```

詳細については、「QBE (例示による問い合わせ)、要約、クラスタリングの有効化」(25 ページ) を参照してください。

QBE を使用する SQL 文の例については、*$SYBASE/$SYBASE_FTS/sample/scripts* ディレクトリのサンプル・スクリプト *sample_text_queries.sql* を参照してください。

near、near/n

near 演算子は、クエリで指定した項目が含まれ、それらが互いに近い位置にあるドキュメントを選択します (この「近い」は相対的な用語です)。検索ワードが互いに最も近くに現れるドキュメントが、最も高い関連性を持つというランク付けを受けます。

near/*n* 演算子は、項目がどれくらいの遠さまでであればよいかを指定します (*n* の最大値は 1000 です)。次の例では、“raconteur” と “home” というワードが 10 ワード以内に現れるドキュメントが選択されます。

```
select t2.copy
from i_blurbs t1, blurbs t2
where t1.id=t2.id
and t1.index_any = "<near/10>(raconteur, home)"
```

or

詳細については、「and、or」(50 ページ) を参照してください。

phrase

phrase 演算子は、特定のフレーズ (特定の順序で並ぶ 2 つ以上の項目のグループ) が含まれるドキュメントを選択します。各結果は、関連性によってランク付けされます。次の例では、フレーズ “gorilla’s head” が含まれるドキュメントが選択されます。

```
select t1.score, t2.copy
from i_blurbs t1, blurbs t2
where t1.id=t2.id and t1.score > 50
and t1.index_any = "<phrase>(gorilla's head)"
```

paragraph

paragraph 演算子は、指定した検索要素が同じ段落に含まれるドキュメントを選択します。段落内でワードが互いに近くにあるほど、関連性によるランク付けでドキュメントが受け取るスコアは高くなります。次の例は、“text” と “search” というワードが同じ段落内に現れるドキュメントを検索します。

```
select t1.score, t2.copy
from i_blurbs t1, blurbs t2
where t1.id=t2.id and t1.score > 50
and t1.index_any = "<many><paragraph>(text, search)"
```

product

product 演算子は、検索要素それぞれについてドキュメントの score 値を乗算します。ドキュメントのスコアを得るために、拡張型全文検索エンジンは各検索要素についてスコアを計算し、その score を乗算します。次に例を示します。

```
select t1.score, t2.copy
from i_blurbs t1, blurbs t2
where t1.id=t2.id and t1.score > 50
and t1.index_any = "<product>(cat, created)"
```

各検索要素の score 値は 78 です。しかし、項目の score 値は乗算されるので、ドキュメントの score 値は 61 (.78 x.78 =.61(100) = 61) です。

sentence

sentence 演算子は、指定した検索要素が同じ文に含まれるドキュメントを選択します。文の中でワードが互いに近くにあるほど、関連性によるランク付けでドキュメントが受けるスコアは高くなります。次の例は、“tax” と “service” というワードが同じ文の中に現れるドキュメントを検索します。

```
select t1.score, t2.copy
from i_blurbs t1, blurbs t2
where t1.id=t2.id and t1.score > 50
and t1.index_any = "<many><sentence>(tax, service)"
```

stem

stem 演算子は、指定したワードとその語尾変化したものが含まれるドキュメントを検索します。たとえば、ワード “cook” を指定した場合、拡張型全文検索エンジンは、“cooked”、“cooking”、“cooks” などが含まれるドキュメントを示します。結果セットの関連性によるランク付けを行うには、クエリで many 変更子を使用します。詳細については、「演算子変更子」(58 ページ) を参照してください。

次のクエリは、stem 演算子を使用して、ワード “create” の語尾変化したものが含まれるドキュメントを見つけます。すなわち、“create” を語幹として含むワードを見つけます。次に示すように、最初のドキュメントには “create” が完全な語幹ではないワード (“creative”) が含まれていますが、このドキュメントも選択されていることに注意してください。

```
select t1.score, t2.copy
from i_blurbs t1, blurbs t2
where t1.id=t2.id and t1.score > 10
and t1.index_any = "<many><stem>create"
score     copy
-----    --------------------------------------------------
78        Anne Ringer ran away from the circus as a child. A
          university creative writing professor and her family
    . . .
78        If Chastity Locksley didn't exist, this troubled world
          would have created her! Not only did she master the mystic
```

sum

sum 演算子は、各検索要素の score 値を合計します。最大値は 100 です。ドキュメントのスコアを得るために、拡張型全文検索エンジンは各検索要素についてスコアを計算し、その score を合計します。

thesaurus

thesaurus 演算子は、検索要素の同義語が含まれるドキュメントを検索します。たとえば、“dog” というワードを使用して検索を実行すると、その同義語 (“canine”、“pooch”、“pup”、“watchdog” など) を使用しているドキュメントが検索されます。各結果は、関連性によってランク付けされます。

拡張型全文検索エンジンには、デフォルトのシソーラスが用意されています。カスタム・シソーラスも作成できます。詳細については、「カスタム・シソーラスの作成」(32 ページ) を参照してください。

次の例では、thesaurus 演算子を使用して、ワード “crave” の同義語が含まれる結果セットを検索します。最初のドキュメントはワード “want”、2 番目はワード “hunger” が含まれているために選択されています。

```
select t2.copy
from i_blurbs t1, blurbs t2
where t1.id=t2.id
and t1.index_any = "<thesaurus>(crave)"

score     copy
----- ------------------------------------------------------------
78    They asked me to write about myself and my book, so here
      goes:  I started a restaurant called "de Gustibus" with two
    . . .
      of restaurant over another, when what they really want is a
    . . .
78    A chef's chef and a raconteur's raconteur, Reginald
      Blotchet-Halls calls London his second home. "Th' palace
    . . .
      his equal skill in satisfying our perpetual hunger for
    . . .
```

topic

topic 演算子は、指定したトピックで定義されている検索基準を満たすドキュメントを選択します。詳細については、「トピックの作成」(35 ページ) を参照してください。たとえば、次の構文を使用して、トピック “engineering” で定義されている基準を満たすドキュメントを検索します。

```
select t2.copy
from i_blurbs t1, blurbs t2
where t1.id=t2.id
and t1.index_any = "<topic>(engineering)"
```

typo/n

typo/n 演算子は、指定したワードとそれに類似したワードが含まれるように検索を拡大します。オプションの *n* 変数では、検索対象ワードと突き合わせたワードの間の最大誤り数を指定します。たとえば、***typo*** `mouse` は、“house”、“louse”、“moose” などのワードを含むドキュメントを返します。

wildcard

wildcard 演算子を使用すると、検索する項目の一部にワイルドカード文字を使用することができます。表 5-5 は、ワイルドカード文字とその属性を示します。

表 *5-5:* 拡張型全文検索エンジンのワイルドカード文字

文字	機能	構文	検索結果
?	英数字 1 文字を指定する。クエリ内で疑問符を使用するときには、wildcard 演算子を使用する必要はない。疑問符は、セット内 ([ ]) または代替パターン内 ({ }) では無視される。	'?an'	“ran”、“pan”、“can”、“ban”
*	英数字 0 文字以上を指定する。クエリ内でアスタリスクを使用するときには、wildcard 演算子を使用する必要はない。ワイルドカード文字列の最初の文字にはアスタリスクを指定してはならない。アスタリスクはセット内 ([ ]) または代替パターン内 ({ }) では無視される。	'corp*'	“corporate”、“corporation”、“corporal”、“corpulent”
[]	セットの中の任意の 1 文字を指定する。ワード内にセットがある場合は、そのワードを逆さ引用符 (“) で囲む。また、セットの中にはスペースを入れない。	<wildcard> ‘c[auo]t‘	“cat”、“cut”、“cot”
{}	カンマで区切られたパターンの中の 1 つを指定する。ワード内にパターンがある場合は、そのワードを逆さ引用符 (“) で囲む。また、セットの中にはスペースを入れない。	<wildcard> ‘bank{s,er,ing}‘	“banks”、“banker”、“banking”
^	セットに含まれない文字を 1 つ指定する。脱字記号 (^) は、セットを指定する左角カッコ ([) に続く最初の文字にする必要がある。	<wildcard> ‘st[^oa]ck‘	“stock” と “stack” を除外し、“stick” と “stuck” を検索する
-	セット内の文字の範囲を指定する。	<wildcard> ‘c[a-r]t‘	“cat” から “crt” までの 3 文字のワードすべて

結果セットの関連性によるランク付けを行うには、クエリで many 変更子を使用します。詳細については、「演算子変更子」(58 ページ) を参照してください。

たとえば次のクエリは、ワード “slingshot” の語尾変化したものが含まれるドキュメントを検索します。

```
select t2.copy
from i_blurbs t1, blurbs t2
where t1.id=t2.id
and t1.index_any = '"slingshot*"'
score    copy
-----    ------------------------------------------------------------
100    Albert Ringer was born in a trunk to circus parents, but
    another kind of circus trunk played a more important role
    . . .
    gorilla.  "Slingshotting" himself from the ring ropes,
    . . .
```

word

word 演算子は、指定したワードを含むドキュメントを検索します。結果セットの関連性によるランク付けを行うには、クエリで many 演算子を使用します。次の例は、blurbs テーブルで、ワード “palates” を含むドキュメントを検索します。

```
select t1.score, t2.copy
from i_blurbs t1, blurbs t2
where t1.id=t2.id and t1.score > 50
and t1.index_any = "<many><word>(palates)"
```

yesno

yesno 演算子は、0 以外の score 値をすべて 100 に変換します。たとえば、5 つのドキュメントの score 値が 86、45、89、89、100 の場合、これらのドキュメントはそれぞれ score 値 100 で返されます。score 値が 0 の場合は 0 のままです。yesno 演算子は、ソート順に関係なく、検索基準に一致したすべてのドキュメントが結果セットに返されたことを確認するのに役立ちます。

演算子変更子

Verity クエリ言語には、検索の精度を高めるために演算子と組み合わせて使用できる変更子があります。表 5-6 で、変更子について説明します。

表 *5-6: Verity* 演算子変更子

変更子名	説明	組み合わせて使用できる演算子	例
case	大文字と小文字を区別して検索する。検索する用語に大文字と小文字が混在している場合、自動的に大文字と小文字が区別される。	wildcard word	`<case><word>(Net)`
many	ワード、語幹のワード、またはフレーズがドキュメントに現れる回数をカウントする。その密度に応じて、ドキュメントに対して関連性によるランク付けを行う。	paragraph phrase sentence stem word wildcard	`<many><stem>(write)`
not	クエリが検索する項目を含むドキュメントを除外する。	and or	`cat<and><not>elephant`

order	ドキュメント内の項目が、クエリと同じ順序で現れることを指定する。
	order 変更子は必ず演算子の直前で指定する。
		near/n
		paragraph
		sentence
			単純な構文：
			`tidbits<order><paragraph>king`
			明示的な構文：
			`<order><paragraph>(tidbits,king)`

第 6 章

システム管理

この章では、拡張型全文検索エンジンに関するシステム管理の問題について説明します。

UNIX での拡張型全文検索エンジンの起動

UNIX で拡張型全文検索エンジンを起動するには、startserver ユーティリティを使用します。startserver ユーティリティは、Adaptive Server の *install* ディレクトリにあります。たとえば、KRAZYKAT という名前の拡張型全文検索エンジンを起動する場合、次のように入力します。

```
startserver -f
$SYBASE/$SYBASE_FTS/install/RUN_KRAZYKAT
```

ここで、-f フラグは runserver ファイルの相対パスを指定しています。このコマンドを発行すると、拡張型全文検索エンジンは設定パラメータの設定内容を表す一連のメッセージを発行します。

runserver ファイルの作成

runserver ファイルには、拡張型全文検索エンジンの起動コマンドが記述されています。*runserver* ファイルには、次のフラグを指定することができます。

- -S*server_name* − 拡張型全文検索エンジンの名前を指定する。interfaces ファイルで設定ファイルとネットワーク接続情報の位置付けに使用する。
- -t − 拡張型全文検索エンジンが、起動メッセージを標準エラーに書き込むようにする。
- -l*errorlog_path* − エラー・ログ・ファイルのパスを指定する。
- -i*interfaces_file_path* − interfaces ファイルのパスを指定する。

サンプルの runserver ファイルはインストール中に *$SYBASE/$SYBASE_FTS/install* ディレクトリにコピーされます。このファイルのコピーを作成し、名前を *RUN_server_name* に変更します。ここで、*server_name* は拡張型全文検索エンジンの名前です。runserver ファイルに、お使いのプラットフォームに対する正しいパス環境変数を指定してください。表 6-1 は、各プラットフォームに対して使用するパス環境変数を示します。

表 *6-1: runserver* ファイル用のパス環境変数

プラットフォーム	環境変数
RS/6000 AIX	LIBPATH
Sun Solaris	LD_LIBRARY_PATH
HP 9000(800)	SHLIB_PATH
Compaq Tru64	LD_LIBRARY_PATH
Linux	LD_LIBRARY_PATH
NT	PATH

たとえば、KRAZYKAT という拡張型全文検索エンジンに対する Sun Solaris 上の runserver ファイルは *RUN_KRAZYKAT* であり、次のような形式になります。

```
#!/bin/sh
#

LD_LIBRARY_PATH="$SYBASE/$SYBASE_FTS/lib:$LD_
LIBRARY_PATH"
export LD_LIBRARY_PATH

$SYBASE/bin/txtsvr -SKRAZYKAT
```

runserver ファイルの起動コマンドは必ず 1 行で記述し、改行を入れないでください。ファイルの内容を 2、3 行にわたって記述する必要がある場合は、改行の代わりに円記号 (¥) を入れます。

Windows NT での拡張型全文検索エンジンの起動

拡張型全文検索エンジンは、Sybase Central™ からサービスとして起動したり、コマンド・ラインから起動したりできます。

- Sybase Central から起動する – サーバの起動については、お使いの Sybase Central のマニュアルを参照してください。
- サービスとして起動する –「サービスとしての拡張型全文検索エンジンの起動」(61 ページ) を参照してください。
- コマンド・ラインから起動する – 次の構文を使用します。

  %SYBASE%¥%SYBASE_FTS%¥bin¥txtsvr.exe -S*server_name*
  [-t] [-i%SYBASE%*path_to_sql.ini_file*] [-l%SYBASE%*path_to_errorlog*]

各パラメータの意味は、次のとおりです。

- -S は、起動する拡張型全文検索エンジンの名前です。
- -t は起動メッセージを標準エラーに書き込みます。
- -i は *sql.ini* ファイルのパスです。このパスには、*sql.ini* ファイル名を含める必要があります。
- -l はエラー・ログのパスです。このパスには、エラー・ログのファイル名を含める必要があります。

たとえば、デフォルトの *sql.ini* ファイル、エラー・ログ・ファイルを使用して、また、起動メッセージをトレースするために -t を使用して、Windows NT 上の KRAZYKAT という拡張型全文検索エンジンを起動するには、次のように入力します。

```
%SYBASE%¥%SYBASE_FTS%¥bin¥txtsvr.exe -SKRAZYKAT -t
```

起動完了メッセージが表示された時点で、拡張型全文検索エンジンが実行状態になります。

サービスとしての拡張型全文検索エンジンの起動

Sybase Central の instsvr ユーティリティを使用して、サービス・ユーティリティによって開始と停止が可能なサービスのリストに拡張型全文検索エンジンを追加します。instsvr は、*%SYBASE%¥%SYBASE_FTS%¥bin* ディレクトリにあります。

instsvr ユーティリティの構文は次のとおりです。

```
instsvr.exe service_name %SYBASE%¥%SYBASE_FTS%¥bin¥txtsvr.exe "startup_parameters"
```

各パラメータの意味は、次のとおりです。

- *service_name* − サービスとして追加する拡張型全文検索エンジンの名前です。Sybase Central では、拡張子が "_TS" のサーバ名 (たとえば KRAZYKAT_TS) を使用することをおすすめします。
- *startup_parameters* − 起動時に使用するパラメータです。

たとえば、KRAZYKAT_TS という名前の拡張型全文検索エンジンをサービスとしてインストールするには、次のように入力します。

```
instsvr.exe KRAZYKAT_TS %SYBASE%¥sds¥text¥bin¥txtsvr.exe
"-SKRAZYKAT_TS -t"
```

注意 1 つのパラメータ (たとえば -i) だけでなく、複数のパラメータを指定するには、すべてのパラメータを 1 組の二重引用符で囲んでください。

拡張型全文検索エンジンの起動と停止を行うように Sybase Central を設定するには、“SYBTXT_*server_name*” で始まるサービス名を指定してください。ここで、*server_name* は interfaces ファイルにリストされた拡張型全文検索エンジンの名前です。たとえば、interfaces ファイル内の名前が KRAZYKAT_TS の場合は、次の instsvr コマンドを実行して、Sybase Central で管理できるサービスを作成します。

```
instsvr SYBTXT_KRAZYKAT_TS %SYBASE%¥%SYBASE_FTS%¥bin¥txtsvr.exe
"-SKRAZYKAT_TS -t"
```

拡張型全文検索エンジンの停止

次のコマンドを使用して、Adaptive Server から拡張型全文検索エンジンを停止します。

```
server_name...sp_shutdown
```

server_name は、停止する拡張型全文検索エンジンの名前です。

拡張型全文検索エンジンを停止できるのは sa_role を持つユーザのみです。

たとえば、KRAZYKAT という名前の拡張型全文検索エンジンを停止するには、次のように入力します。

```
KRAZYKAT...sp_shutdown
```

設定パラメータの修正

各拡張型全文検索エンジンにはデフォルト値を持つ設定パラメータがあります。表 6-2 に、これらのパラメータを示します。

表 6-2: 設定パラメータ

パラメータ	説明	デフォルト値
batch_size	拡張型全文検索エンジンに送信されるバッチのサイズを決定する。	500
batch_blocksize	このパラメータが有効な場合、Text Server はデータをより小さなまとまりで読み込む。このパラメータは、一度に *n* 個のローを検索するように Text Server に指示する。0 (無効) ～ 65535 の範囲で指定する。	0
max_indexes	拡張型全文検索エンジン内で作成されるテキスト・インデックスの最大数。	126
max_stacksize	クライアント・スレッドに割り付けられるスタックのサイズ (キロバイト単位)。	34,816

パラメータ	説明	デフォルト値
max_threads	拡張型全文検索エンジンで利用できるスレッドの最大数。	50
max_packetsize	拡張型全文検索エンジンと Adaptive Server 間で送信されるパケットのサイズ。	2048
max_sessions	拡張型全文検索エンジンのセッションの最大数。	100
min_sessions	拡張型全文検索エンジンのセッションの最小数。	10
language	拡張型全文検索エンジンで使用する言語。	us_english
charset	拡張型全文検索エンジンで使用する文字セット。	iso_1
vdkCharset	Verity 検索エンジンで使用する文字セット。	850
vdkLanguage	Verity 検索エンジンで使用する言語。	english
vdkHome	Verity ディレクトリ。	UNIX の場合： $SYBASE/$SYBASE_FTS/verity Windows NT の場合： *%SYBASE%¥%SYBASE_FTS%¥verity*
collDir	拡張型全文検索エンジンのコレクションの格納領域のロケーション。	UNIX の場合： *$SYBASE/$SYBASE_FTS/collections* Windows NT の場合： *%SYBASE%¥%SYBASE_FTS%¥collections*
defaultDb	テキスト・インデックス・メタデータを格納する拡張型全文検索エンジン・データベースの名前。	text_db
interfaces	拡張型全文検索エンジンが使用する interfaces ファイルがあるディレクトリへのフル・パス。	UNIX の場合： *$SYBASE/interfaces* Windows NT の場合： *%SYBASE%¥ini¥sql.ini*
sort_order	デフォルトのソート順。	0
errorLog	エラー・ログ・ファイルへのフル・パス名。	拡張型全文検索エンジンを起動するディレクトリ。
traceflags	診断情報を生成するための数値フラグ識別子を表す文字列。	0
srv_traceflags	Open Server の診断情報を生成するための数値フラグ識別子を表す文字列。	0
max_session_fd	拡張型全文検索セッションで使用されるファイル記述子の最大数。詳細については、「ファイル記述子と拡張型全文検索」(81 ページ) を参照。	0
cluster_style	使用するクラスタ化のスタイル。	Fixed
cluster_max	cluster_style が Fixed に設定されているときに生成するクラスタの最大数。	0
cluster_effort	拡張型全文検索エンジンが適正なクラスタリングに費やす時間。	Default
cluster_order	クラスタ内のクラスタとローを返す順序。	0

パラメータ	説明	デフォルト値
auto_online	拡張型全文検索エンジンの起動時にインデックスを自動的にオンラインにするかどうかを指定する。インデックスを自動的にオンラインにする場合は 1 を設定し、それ以外の場合は 0 を設定する。	0
backDir	テキスト・インデックスのバックアップ・ファイルを格納するデフォルトのロケーション。	UNIX の場合： *$SYBASE/$SYBASE_FTS/backup* Windows NT の場合： *%SYBASE%¥%SYBASE_FTS%¥backup*
knowledge_base	Verity のトピック機能を実現するための知識ベース・マップのロケーション。	NULL
nocase	拡張型全文検索エンジンの大文字と小文字の区別を設定する。Adaptive Server のソート順序で大文字と小文字を区別する場合は nocase を 0 に設定し、区別しない場合は 1 に設定する。	0

これらのパラメータがすべて入っているサンプル設定ファイルは、インストール中にインストール・ディレクトリにコピーされます。サンプル設定ファイルは、*textsvr.cfg* という名前であり、「付録 B サンプル・ファイル」にあります。

設定値の修正

設定パラメータの値を修正するには、sp_text_configure を使用します。構文は次のとおりです。

sp_text_configure *server_name*, *config_name*, *config_value*

各パラメータの意味は、次のとおりです。

- *server_name* – 拡張型全文検索エンジンの名前です。
- *config_name* – 設定パラメータの名前です。
- *config_value* – 設定パラメータに割り当てる値です。

詳細については、sp_text_configure (143 ページ) を参照してください。

利用可能な設定パラメータ

表 6-3 は、利用可能な設定パラメータとその値の制限を示します。

表 *6-3:* 設定パラメータの制限

パラメータ	値	静的 / 動的
batch_size	0 ～ MAX_INT	動的
batch_blocksize	0 ～ 65535	動的
max_indexes	0 ～ MAX_INT	静的
max_stacksize	0 ～ MAX_INT	静的
max_threads	0 ～ MAX_INT	静的
max_packetsize	0 ～ MAX_INT	静的
max_sessions	0 ～ MAX_INT	静的
min_sessions	0 ～ max_sessions	静的
language	french、spanish、german、us_english	静的
charset	ascii_8、cp037、cp1047、cp437、cp500、cp850、deckanji、eucjis、iso_1、mac、roman8、sjis、utf8	静的
vdkCharset	50、437、1252、mac1 (マニュアルにリストされているもののみ)	静的
vdkLanguage	frenchx、spanishx、germanx、english、englishx、bokmalx、dutchx、finnishx、nynorskx、swedishx、portugx、italianx、danishx	静的
vdkHome	1 つの文字列 < 255 文字	静的
collDir	1 つの文字列 < 255 文字	静的
default_Db	1 つの文字列 < 32 文字	静的
interfaces	1 つの文字列 < 255 文字	静的
sort_order	0、1、2、3	動的
errorLog	1 つの文字列 < 255 文字	静的
traceflags	カンマで区切った 1 ～ 15 までの任意の数字を含む文字列	静的
srv_traceflags	カンマで区切った 1 ～ 8 までの任意の数字を含む文字列	静的
cluster_style	Coarse、Medium、Fine、Fixed	動的
cluster_max	0 ～ MAX_INT	動的
cluster_effort	Low、Medium、High、Default	動的
cluster_order	0 または 1	動的
auto_online	0 または 1	静的
backCmd	1 つの文字列 < 255 文字	動的
restoreCmd	1 つの文字列 < 255 文字	動的
backDir	1 つの文字列 < 255 文字	静的
knowledge_base	1 つの文字列 < 255 文字	静的
nocase	0 または 1	動的
max_session_fd	0、5 ～ MAX_INT	静的

デフォルト言語の設定

Verity のデフォルト言語は、vdkLanguage 設定パラメータを使用して設定します。デフォルトでは、vdkLanguage は “english” に設定されていますが、異なるデフォルト言語を使用するように設定することもできます。表 6-4 に、Sybase でサポートされるロケールのリストを示します。

表 *6-4: vdkLanguage* 設定パラメータ

言語	デフォルトのロケール名
英語	english
ドイツ語	german
フランス語	french

$SYBASE/$SYBASE_FTS/verity/common ディレクトリにはその他の言語アダプタもありますが、拡張型全文検索エンジンから表示されるメッセージは表 6-4 の言語のみです。

language パラメータは、拡張型全文検索エンジンによるエラー・メッセージ、および Open Server、Open Client のエラー・メッセージの表示言語です。language パラメータは Adaptive Server と同じ言語に設定してください。

拡張型全文検索エンジンで、次のように入力します。

```
sp_text_configure KRAZYKAT, 'vdkLanguage', 'spanish'
```

Verity 言語の詳細については、Verity Web サイト (http://www.verity.com) を参照してください。

デフォルトの文字セットの設定

Verity のデフォルトの文字セットは、設定ファイルに vdkCharset パラメータを指定して設定します。Verity の文字セットで使用するファイルは *$SYBASE/$SYBASE_FTS/verity/common* にあります。表 6-5 に、Verity で使用できる文字セットを示します。

表 *6-5: Verity* の文字セット

文字セット	説明
850	デフォルト
437	IBM PC 文字セット
1252	西欧言語用 Windows コード・ページ
mac1	Macintosh Roman

拡張型全文検索エンジンのデフォルトの文字セットは、charset パラメータを使用して設定します。charset パラメータは Adaptive Server の文字セットに合わせて設定してください。

たとえば、拡張型全文検索エンジンで、次のように入力します。

```
sp_text_configure KRAZYKAT, 'vdkCharset', '437'
```

ユーロ記号のインデックス作成

ユーロ記号のインデックスを作成するには、Adaptive Server に utf8 文字セットをインストールします。vdkLanguage を <language>x に設定し、vdkCharset をブランクにします。次に例を示します。

```
ASE 12.5.x charset       =   utf8
EFTS 12.5.x vdkLanguage  =   englishx
EFTS 12.5.x vdkCharset   =
```

デフォルトのソート順の設定

デフォルトでは、拡張型全文検索エンジンは結果セットを score 疑似カラムを降順に (スコアの高い方が先になるように) ソートします。デフォルトのソート順を変更するには、sort_order 設定パラメータを表 6-6 に示した値のいずれかに設定します。

表 *6-6:* 設定ファイルのソート順の値

値	説明
0	score 疑似カラムを降順にソートして結果セットを返す。デフォルト値。
1	score 疑似カラムを昇順にソートして結果セットを返す。
2	タイムスタンプを降順にソートして結果セットを返す。
3	タイムスタンプを昇順にソートして結果セットを返す。

たとえば、次のように入力します。

```
sp_text_configure KRAZYKAT, 'sort_order', '2'
```

タイムスタンプの降順 (表 6-6 の値 2) で結果セットをソートすると、拡張型全文検索エンジンは最新のドキュメントを最初に返します。最新のドキュメントとは、最後に挿入または更新されたドキュメントです。タイムスタンプの昇順 (表 6-6 の値 3) で結果をソートすると、拡張型全文検索エンジンは最も古いドキュメントを最初に返します。

デフォルトのソート順の設定は、クエリで max_docs 疑似カラムを使用する場合に特に重要です。max_docs 疑似カラムによって、結果セットのローの数が、ソート順に配列された最初の *n* 個のローに制限されます。max_docs を結果セットのサイズより小さい数に設定している場合、選択したソート順で、検索している情報を含むローが除外されることもあります。

たとえば、タイムスタンプの昇順でソートした場合、テーブルに追加された最新のドキュメントが結果セットの最後になります。結果セット全体で 11 個のドキュメントがあるときに max_docs が 10 に設定されている場合、最新のドキュメントは結果セットに入りません。ただし、タイムスタンプの降順でソートした場合は、最新のドキュメントが結果セットの最初になります。

トレース・フラグの設定

traceflags パラメータを使用すると、拡張型全文検索エンジンで発生した特定のイベントのロギングが可能になります。各トレース・フラグは、番号によってユニークに識別されます。表 6-7 に、トレース・フラグの説明を示します。

表 *6-7:* 拡張型全文検索エンジンのトレース・フラグ

トレース・フラグ	説明
1	Adaptive Server の接続／切断イベントとアテンション・イベントをトレースする。
2	言語イベントをトレースする。Adaptive Server によって拡張型全文検索エンジンに送信された SQL 文をトレースする。
3	RPC イベントをトレースする。
4	カーソル・イベントをトレースする。Adaptive Server によって拡張型全文検索エンジンに送信された SQL 文をトレースする。
5	表示されるエラーをログに書き込む。
6	テキスト・インデックスに関する情報をトレースする。Verity に渡される検索文字列をログに書き込み、検索によって返されるレコードの数をログに書き込む。
7	送信済みパケットをトレースする。
8	拡張型全文検索エンジンと Verity API 間のインタフェースに対する呼び出しをトレースする。
9	SQL 解析をトレースする。
10	Verity 処理をトレースする。
11	Verity コレクションの最適化を無効にする。
12	sp_statistics から情報が返されないようにする。
13	バックアップ処理をトレースする。
14	Verity のステータス情報とタイミング情報を記録する。
15	コレクションの ngram インデックス情報を生成する。ngram を使用するとワイルドカード検索を高速実行できる。このトレース・フラグは、Unicode フォーマットのデータに対するワイルドカード検索で必要である。
30	このトレース・フラグは、古いコレクション・ファイルを削除する。Verity MaxClean 機能を有効にする。追加の処理時間を費やし、通常の使用の妨げになる場合があるため、管理時だけに使用する。sp_optimize_text_index と一緒に使用すると有効になる。

トレース・フラグを対話的に有効にしたり無効にしたりするには、拡張型全文検索エンジンで RPC (リモート・プロシージャ・コール) の sp_traceon または sp_traceoff を使用します。

sp_traceon を実行するには、次の構文を使用します。ここで、*textserver* は、拡張型全文検索エンジンの名前です。

textserver...sp_traceon 1,2,3,4

traceflag は、セッションが終了するまで、または sp_traceoff RPC が特定の traceflag を使用して実行されるまで、アクティブです。traceflag を永続的に設定するには、traceflag を設定ファイルに設定するか、または sp_text_configure コマンドを使用します。

Open Server のトレース・フラグの設定

srv_traceflags パラメータを使用して、Open Server の診断情報を記録するトレース・フラグをオンにします。表 6-8 に、Open Server のトレース・フラグの説明を示します。

表 *6-8: Open Server* のトレース・フラグ

トレース・フラグ	説明
1	TDS ヘッダをトレースする。
2	TDS データをトレースする。
3	アテンション・イベントをトレースする。
4	メッセージ・キューをトレースする。
5	TDS トークンをトレースする。
6	Open Server イベントをトレースする。
7	遅延イベント・キューをトレースする。
8	ネットワーク要求をトレースする。

次に例を示します。

```
sp_text_configure KRAZYKAT, 'srv_traceflags', '3'
```

大文字と小文字の区別の設定

デフォルトでは、拡張型全文検索エンジンは大文字と小文字を区別します。つまり、識別子は大文字と小文字が正確に入力されないと認識されません。たとえば、blurbs (小文字) というテーブルがあった場合、テーブル名に BLURBS と指定した sp_create_text_index コマンドを発行することはできません。コマンドを発行するときは、テーブル名引数には大文字と小文字まで一致したコマンドを発行してください。

```
sp_create_text_index "KRAZYKAT", "i_blurbs", "blurbs", "", "copy"
```

拡張型全文検索エンジンで、nocase パラメータを使用すると、拡張型全文検索エンジンの大文字と小文字の区別を設定できます。0 の場合は大文字と小文字が区別され、1 の場合は区別されません。nocase パラメータは、Adaptive Server でのソート順における大文字と小文字の区別に合わせて設定してください。

次の例では、大文字と小文字を区別しないように KRAZYKAT サーバを変更します。

```
sp_text_configure KRAZYKAT, 'nocase', '1'
```

注意 nocase パラメータは、Verity クエリの大文字と小文字の区別には影響しません。詳細については、「Verity 演算子を使用するときの考慮事項」(48 ページ) を参照してください。

拡張型全文検索エンジンでのバックアップとリカバリ

Enhanced Full-Text Search Specialty Data Store のバックアップとリカバリは、sp_text_dump_database と sp_text_load_index によって自動で行われます。これらのシステム・プロシージャによって、データとテキスト・インデックスの整合性を維持するためのシームレスなインタフェースが提供されます。

Adaptive Server のユーザ・データベースと Verity コレクションは物理的に別個のものとして存在します。ユーザ・データベースをバックアップしても、Verity コレクションがバックアップされるわけではなく、バックアップからデータベースをリストアしても、Verity コレクションがリストアされるわけではありません。『システム管理ガイド』の「第 27 章 ユーザ・データベースのバックアップとリストア」に説明されているバックアップ・プロシージャとリカバリ・プロシージャは、Adaptive Server のユーザ・データベースおよび text_db データベースにのみ適用されます。

『システム管理ガイド』の「第 26 章 バックアップおよびリカバリ・プランの作成」に説明されている、データベースに対する推奨のバックアップ・スケジュールに従ってください。Sybase では、テキスト・インデックスのあるユーザ・データベースをバックアップするときに次のものもバックアップすることをおすすめします。

- text_db データベース
- テキスト・インデックス

注意 テキスト・インデックスをリカバリするために、ユーザ・データベースとテキスト・インデックスを同時にバックアップする必要はありません。しかし、ユーザ・データベースをリストアしてからテキスト・インデックスをリストアしてください。これにより text_events テーブルがリストアされます。このテーブルは、テキスト・インデックスをユーザ・データベースと同期させるために sp_text_load_index が使用します。

定期的なバックアップ・スケジュールにより、テキスト・インデックス、Adaptive Server データ、text_events テーブルの整合性が保証されます。テキスト・インデックスの削除と再作成を行わないでリカバリするには、これらがすべて必要になります。

カスタマイズ可能なバックアップとリストア

backCmd と restoreCmd を使用すると、コレクション・ファイルをバックアップするときに、カスタマイズ可能なバックアップ・コマンドとリストア・コマンドを tar コマンドまたは zip コマンドの代わりに使用できます。backCmd と restoreCmd がブランクの場合はデフォルトのコマンドが使用され、ブランクでない場合は指定したコマンドが実行されます。実行の前に文字列の置換が行われるため、入出力ディレクトリとコレクション ID の指定が可能になります。文字列の置換は次のように定義されています。

- ${backDir} は、“backDir” 設定パラメータとして指定された *backup* ディレクトリによって置き換えられる
- ${collDir} は、コレクションのフル・パス名によって置き換えられる
- ${colID} は、バックアップ・ファイルのフル・ネームであるコレクション ID によって置き換えられる

Verity コレクションのバックアップ

sp_text_dump_database システム・プロシージャは、コレクションのバックアップ、およびユーザ・データベースと text_db データベースのバックアップ (オプション) を行います。sp_text_dump_database はまた、リカバリの必要がなくなったエントリを削除することにより、text_events テーブルの管理も行います。

バックアップ中、拡張型全文検索エンジンはクエリを処理しますが、バックアップが完了するまで更新要求を延期します。このため、拡張型全文検索エンジンを停止して再起動する必要はありません。

バックアップするテキスト・インデックスを含むデータベースに対して sp_text_dump_database を実行します。sp_text_dump_database を発行するときは、必要なサーバがすべて稼働していることを確認します。sp_text_dump_database はすべての拡張型テキスト・サーバのすべてのインデックスを無条件でバックアップします。テキスト・インデックスのバックアップは、backDir 設定パラメータに指定されたディレクトリに置かれます。dump database の出力は、拡張型全文検索のエラー・ログに書き込まれます。テキスト・インデックスがバックアップされるときに現在のデータベースと text_db データベースをダンプすることをおすすめします。ただし、このダンプは任意です。

たとえば、テキスト・インデックスとデータベースをバックアップする場合で、sample_colors_db データベースを */work2/sybase/colorsbackup* ディレクトリに、text_db データベースを */work2/sybase/textdbbackup* ディレクトリにバックアップするには、次のように入力します。

```
sp_text_dump_database @backupdbs =
INDEXES_AND_DATABASES, @current_to = "to
'/work2/sybase/colorsbackup'", @textdb_to="to
'/work2/sybase/textdbbackkup'"
```

注意 テキスト･インデックスをバックアップするたびに text_db データベースをバックアップすることが重要です。このデータベースにはすべてのテキスト・インデックスのメタデータが入っているためです。

必要なファイル・サイズが 2GB より大きい場合、Solaris では sp_text_dump_database でエラーが発生する可能性があります。

詳細については、「sp_text_dump_database」(144 ページ) を参照してください。

バックアップからのコレクションとテキスト・インデックスのリストア

sp_text_load_index システム・プロシージャは、sp_text_dump_database システム・プロシージャによってバックアップされたテキスト・インデックスをリストアします。

データベース管理者は、次のプロシージャを実行して Verity コレクションをリストアしてください。

1 Adaptive Server のユーザ・データベースと text_db データベースをリストアします。これにより、送信元テーブル、メタデータ、text_events テーブルが、一貫した予測できる状態に戻ります。『システム管理ガイド』の「第 27 章 ユーザ・データベースのバックアップとリストア」に説明されているプロシージャに従い、ユーザ・データベースと text_db データベースをリストアします。

2　sp_text_load_index を実行して、最新のインデックス・ダンプから Verity コレクションをリストアします。このプロシージャは、最新のインデックスのダンプ以降に作成されたすべての text_events テーブル・エントリのステータスを “unprocessed” にリセットし、拡張型全文検索エンジンにこれらのイベントを処理するように通知します。

例：

sample_colors_db データベースとそのすべてのテキスト・インデックスをリストアするには、次の手順に従います。

1　text_db データベースをリストアします。

```
1> use master
2> go
1> load database text_db from '/work2/sybase/textdbbackkup'
2> go
```

2　sample_colors_db データベースをリストアします。

```
1> load database sample_colors_db from '/work2/sybase/colorsbackup'
2> go
```

3　text_db データベースと sample_colors_db データベースをオンラインにします。

```
1> online database text_db
2> online database sample_colors_db
3> go
```

4　テキスト・インデックスをリストアします。

```
1> use sample_colors_db
2> go
1> sp_text_load_index
2> go
```

詳細については、「sp_text_load_index」(147 ページ) を参照してください。

第 7 章

パフォーマンスとチューニング

拡張型全文検索エンジンの設定は、出荷時にはデフォルトの設定になっています。このデフォルトの設定を変更して、サイトにおけるニーズをさらに反映できるように、拡張型全文検索エンジンのパフォーマンスを最適化できます。この章では、パフォーマンスを強化する方法について説明します。

既存のインデックスの更新

次の手順に従い、トレース・フラグ 11、トレース・フラグ 12、またはその両方を有効 (on) にすると、テキスト・インデックス内のレコードの更新時間を短縮できます。

- トレース・フラグ 11 を有効にすると、Verity コレクションの最適化が無効になります。つまり、Verity によるテキスト・インデックスの最適化が行われないので、sp_text_notify を発行するとパフォーマンスが向上します。トレース・フラグ 11 が off (デフォルト) の場合は、sp_text_notify 処理の最後に、拡張型全文検索エンジンに呼び出された Verity によってテキスト・インデックスが最適化されます。このため、sp_text_notify の完了が遅くなることがあります。

  Enhanced Full-Text Search Specialty Data Store を使用する場合、トレース・フラグ 11 を有効にしておくと、後からでもテキスト・インデックスを、sp_optimize_text_index を使用して最適化できます。詳細については、「sp_optimize_text_index」(137 ページ) を参照してください。

- トレース・フラグ 12 を有効にすると、拡張型全文検索エンジンから sp_statistics 情報が返されなくなります。トレース・フラグ 12 が off (デフォルト) の場合は、拡張型全文検索エンジンに update statistics コマンドが発行されるため、sp_text_notify の完了が遅くなることがあります。

テキスト・インデックスを数秒ごとに更新するほど更新頻度が高い場合は、update statistics 処理、Verity の最適化、またはその両方を無効にすると、ほとんどの更新についてパフォーマンスを向上させることができます。

トレース・フラグ 11 と 12 は、拡張型全文検索エンジン内で sp_traceon と sp_traceoff を使用することで、対話的に有効にしたり無効にしたりできます。

クエリのパフォーマンスの向上

クエリのパフォーマンスを著しく向上させるには、次の 2 つの方法があります。

- 拡張型全文検索エンジンから返されるローの数を制限する
- クエリのジョイン順の適正化

ローの数の制限

max_docs 疑似カラムを使用して、拡張型全文検索エンジンから返されるローの数を制限します。拡張型全文検索エンジンから返されるローの数が少ないほど、Adaptive Server は送信元テーブルとインデックス・テーブルの間でジョインを速く処理できます。

クエリのジョイン順の適正化

テーブルとテキスト・インデックスをジョイン内に多くリストするほど、不正なジョイン順によるクエリの実行遅延の確率が増大します。テキスト・インデックスと 1 つ以上のテーブルの間でジョインを実行中の場合は、テキスト・インデックスを最初に問い合わせると、クエリが最も速く実行されます。

ジョイン順を正しくするには、次の手順に従います。

- インデックスを作成しているテーブルの IDENTITY カラムに、ユニークなクラスタード・インデックスまたはユニークなノンクラスタード・インデックスが作成されていることを確認する
- ジョインの対象を 1 つのベース・テーブルと 1 つのテキスト・インデックスに制限する

クエリの実行が遅い場合は、showplan を使用するか、トレース・フラグ 11205 を有効にして、ジョイン順を検査します。トレース・フラグ 11205 によって、リモート・クエリは Adaptive Server のエラー・ログ・ファイルにダンプされます。クエリを最速にするには、where 句の中で検索条件に index_any を指定し、最初にテキスト・インデックスに対する問い合わせが実行されるようにします。

クエリの実行が最も遅いのは、where 句にテキスト・インデックスの id カラムが指定されている場合で、最初にインデックス・テーブルに対する問い合わせが実行されます。この場合、クエリを書き換えるか、forceplan を使用して、クエリ内にリストされているジョイン順を強制的に指定します。forceplan の詳細については、『パフォーマンス&チューニング・シリーズ：クエリ処理と抽象プラン』の「第 12 章 抽象プランの作成と使用」を参照してください。

Adaptive Server の再設定

この項で説明する Adaptive Server 設定パラメータをリセットすることで、拡張型全文検索エンジンのパフォーマンスを向上させることができます (sp_configure を使用して設定パラメータを設定する方法については、『システム管理ガイド』の「第 4 章 設定パラメータ」を参照してください)。

cis cursor rows

cis cursor rows パラメータを使用して、Adaptive Server が単一のフェッチ・オペレーションの間に受信するローの数を指定します。デフォルトでは、cis cursor rows は 50 です。この数を増やすと、Adaptive Server がフェッチ・オペレーション中に拡張型全文検索エンジンから受信するローの数が増えます。ただし、cis cursor rows の値を大きくすると、Adaptive Server が結果セットを返すために割り付ける動的メモリも増大します。

cis packet size

cis packet size は、単一のネットワーク・パケットを構成するバイト数を決定します。cis packet size のデフォルトは 512 です。このパラメータの値は 512 の倍数にしてください。パケット・サイズが大きいと各クエリに返すパケット数が少なくなるので、このパラメータの値を大きくすると拡張型全文検索エンジンのパフォーマンスが向上します。ただし、cis packet size の値を大きくすると、Adaptive Server がこのパラメータに対して割り付けるメモリも増大します。

cis packet size は動的なパラメータであり、変更内容がすぐに反映されるため、Adaptive Server を再起動する必要はありません。

注意 cis packet size を変更した場合は、拡張型全文検索エンジン設定ファイル内の max_packetsize も同じ値に変更してください。コンポーネント統合サービスを使用してほかのリモート・サーバにもアクセスしている場合は、そのサーバのネットワーク・パケットの最大サイズも同様に増やしてください。

---

max_packetsize パラメータを有効にするには拡張型全文検索エンジンを再起動する必要があります。

拡張型全文検索エンジンの再設定

この項で説明する拡張型全文検索エンジン設定パラメータを再設定すると、拡張型全文検索エンジンのパフォーマンスを向上させることができます。詳細については、「設定パラメータの修正」(62 ページ) を参照してください。

batch_size

batch_size によって、拡張型全文検索エンジンが 1 つのバッチごとにインデックスを作成するローの数が決定します。batch_size のデフォルトは 500 です (つまり、1 つのバッチで 500 ローのデータのインデックスが作成されます)。インデックスを作成するバッチのサイズを増やすと、パフォーマンスが向上します。ただし、バッチのサイズを大きくすると、拡張型全文検索エンジンがこのパラメータのために割り付けるメモリも増大します。

batch_size の設定値を検討するときは、テキスト・インデックスを作成するデータのサイズを考慮します。テキスト・インデックスを作成すると、拡張型全文検索エンジンによって、次のサイズ (バイト単位) のメモリが割り付けられます。

(データに必要な領域の大きさ) x (batch_size) = 使用されるメモリ

たとえば、インデックスを作成するデータの大きさが 1 つのローあたり 10,000 バイト、batch_size が 500 に設定されている場合、拡張型全文検索エンジンはテキスト・インデックスの作成時に約 5MB のメモリを割り付ける必要があります。

通常使用するデータのサイズとマシンで使用可能なメモリ量に基づいて、バッチのサイズを選択します。

min_sessions と max_sessions

min_sessions と max_sessions によって、拡張型全文検索エンジンで使用できるユーザ接続の最小数と最大数が決定します。各ユーザ接続には、約 5MB のメモリが必要です。max_sessions の値を、使用可能なメモリ量を超える値に設定しないでください。また、min_sessions 用のメモリは起動時に割り付けられるので、(多数のユーザ接続を許可するために) min_sessions の値を大きくしすぎると、メモリの大部分が拡張型全文検索エンジンのユーザ接続に使用されることになります。

使用するユーザ・セッション数の平均値と同じ値を min_sessions に設定すると、拡張型全文検索エンジンのパフォーマンスが向上します。このように設定すると、ユーザ・セッションを開始するときに、拡張型全文検索エンジンがメモリを割り付けないようにすることができます。

sp_text_notify の使用方法

サイトにおけるニーズを確認してから、sp_text_notify を発行する頻度を決定します。

sp_text_notify はデータを読み込むだけでなくテキスト・コレクションも更新するので、このシステム・プロシージャを使用すると拡張型全文検索エンジンに負荷がかかります。この負荷が大きい場合は、sp_text_notify 発行時のパフォーマンスに大きく影響することがあります。この問題はパフォーマンスと関係するため、インデックスをどの程度の最新状態に保つかを決定する必要があります。インデックスをリアルタイム並みに保つ必要がある場合は、sp_text_notify を頻繁 (5 秒に 1 回程度) に発行します。ただし、インデックスをそこまで新しく保つ必要がない場合は、システムがアクティブでなくなるまで待ってから sp_text_notify を発行すると負荷を抑えられます。

注意　トランザクション内から sp_text_notify を発行することはできません。

複数の拡張型全文検索エンジンの設定

頻繁に使用するテーブルの場合は、テーブルのテキスト・インデックスを別々の拡張型全文検索エンジンに配置すると、パフォーマンスを向上させることができます。ユーザのすべてのクエリを単一の拡張型全文検索エンジンに送信しないで、複数のエンジンに送信してクエリを分散させることができるため、パフォーマンスが向上します。各 Adaptive Server は複数の拡張型全文検索エンジンに接続できますが、各拡張型全文検索エンジンは 1 台の Adaptive Server にしか接続できません。

起動時における複数の拡張型全文検索エンジンの作成

複数の拡張型全文検索エンジンを初めて作成する場合は、installtextserver スクリプトを編集してすべての拡張型全文検索エンジンを指定します。詳細については、「installtextserver スクリプトの編集」(13 ページ) を参照してください。

拡張型全文検索エンジンの追加

最初の起動後に、isql から sp_addserver を発行すると、拡張型全文検索エンジンを追加できます。

```
sp_addserver server_name [, server_class [, physical_name]]
```

各パラメータの意味は、次のとおりです。

- *server_name* は、システム上のサーバを指定するために使用する名前です (この場合、拡張型全文検索エンジン)。
- *server_class* は、追加するサーバのカテゴリを特定します。拡張型全文検索エンジンの場合は、“sds” を指定します。
- *physical_name* は、*server_name* のサーバによって使用される interfaces ファイル内の名前です。

詳細については、『ASE リファレンス・マニュアル』の「sp_addserver」を参照してください。

たとえば、BLUE という名前の拡張型全文検索エンジンを追加する場合、次のように入力します。

```
sp_addserver BLUE, sds, BLUE
```

拡張型全文検索エンジンを設定して起動したら、次の構文を使用して Adaptive Server から拡張型全文検索エンジンに接続できるかどうかを確認できます。

```
server_name...sp_show_text_online
```

たとえば、BLUE というサーバに接続する場合は、次のように入力します。

```
BLUE...sp_show_text_online
```

拡張型全文検索エンジンの追加設定

それぞれの拡張型全文検索エンジンには次のものが必要です。

- interfaces ファイルのエントリ
- 構成ファイル

すべての拡張型全文検索エンジンは、テキスト・インデックスのメタデータを格納するために同じデータベース (デフォルトでは text_db)、および同じ vesaux テーブルと vesauxcol テーブルを使用します。

複数のユーザ

以下のヒントは、複数ユーザでのデッドロックを避けるために役立ちます。

1　Adaptive Server が使用している接続数が拡張型全文検索の数と同じであることを確認します。デフォルトは 100 です。

```
sp_configure "user connections", 100
```

2　次のようにして、(model データベース、または新規データベースごとに存在する) vesaux テーブル、vesauxcol テーブル、text_events テーブルでロー・レベルのロックが使用されていることを確認します。

既存のテーブルの場合：alter table table_name lock datarows

新しいテーブルの場合：create table ... lock datarows

3　多くのコマンドを含む大きなバッチの場合は、いくつかの小さなトランザクションに分割します。

4　依然としてデッドロックが発生する場合は、Adaptive Server が使用できるロックの数を増やし、ローに対するロックの拡大の設定を調整します。詳細については、『システム管理ガイド』を参照してください。

ファイル記述子と拡張型全文検索

拡張型全文検索は、検索の実行時にファイル記述子を広範囲にわたって利用します。同時検索の場合やテキスト・インデックスが大きい場合、これが原因で接続時に次のエラー・メッセージが表示されることがあります。

```
ERRORMSG, Error (): Available files (-1) less than min 5
```

このメッセージは、拡張型全文検索プロセスが、プロセスの制限によってファイル記述子を使い果たしたことを示しています。このエラー・メッセージが表示される場合は、拡張型全文検索プロセスのファイル記述子に対するプロセス制限値を大きくしてください。

max_session_fd は、拡張型全文検索セッションが割り付けることのできるファイル記述子の数を制限します。ファイル記述子の制限値を大きくできない状況ではこれを利用できます。

max_session_fd のデフォルトは 0 であり、各セッションが拡張型全文検索プロセスのファイル記述子の制限値によってのみ制限されることを意味します。max_session_fd の最小設定値は 5 です。

上記のエラー・メッセージが表示され、拡張型全文検索のファイル記述子の制限値を大きくすることができない場合に、((ファイル記述子の最大数) - 20)/(同時接続の最大数) を計算すると、このパラメータの最適値を決定できます。

たとえば、ファイル記述子の制限値が 1024 に設定されており、拡張型全文検索への同時接続の最大数が 50 の場合は、次のようになります。

((1024) - 20)/50 ≒ 20

拡張型全文検索セッションで使用可能なファイル記述子の数が多いほど、この値は大きくなります。

それぞれの拡張型全文検索セッションで使用するファイル拡張子の最大数が制限されるため、パフォーマンスが低下することがあります。Sybase では、max_session_fd を慎重に使用することをおすすめします。

max_session_fd の変更を有効にするには、拡張型全文検索プロセスを再起動する必要があります。

第 8 章

Verity のトピック

この章は、Verity の承諾を得て複製したものです。拡張型全文検索のユーザを対象に、Verity のトピックについて詳しく説明します。

トピックとは

トピックとは、概念、またはサブジェクト領域に関連する情報をグループ化したものです。トピックは、知識を要約し、エンド・ユーザが共有リソースとして利用できるようにする便利な方法です。Verity アプリケーションにトピックを追加することで、ユーザは、トピックが表している内容をより簡単に検索できます。

トピックを組み合わせて構成する知識ベースは、ユーザが検索を行うときに利用できる知識のカタログです。知識ベースによって、ユーザは複雑な構文を使用した高度なクエリを作成せずに、必要な情報を検索することができます。

トピックの編成

トピックは、関連する検索基準のグループをアウトラインのフォーマットと同様のフォーマットに編成します。演算子と変更子は、検索基準の関連グループを結び付ける働きをします。トピックは、独立した単位としても、または階層構造の中で他のトピックへの関連性を持つ単位としても作成できます。

ウェイトの割り当て

検索基準のグループの一部に、同じトピックの構造にある他の検索基準のグループよりも重いウェイトを置くこともできます。検索基準にウェイトを割り当てると、検索で選択されるドキュメントの重要性に影響します。ドキュメントが結果リストで上位にあるほど、そのドキュメントは検索基準に対してより重要であるか、強い関連性があります。検索基準のウェイトは、0.01 から 1.00 までの数値です。選択されたドキュメントの結果リストでの位置から、そのドキュメントと検索基準との関連性をすばやく把握することができます。

トピックのアウトライン・ファイルの使用

トピックを構成するには、トピックのアウトライン・ファイルを作成します。

トピックのアウトライン・ファイルは、トピック定義を含む構造化フォーマットの ASCII テキスト・ファイルです。その一例を以下に示します。

```
$Control:1
art <Accrue>
*performing-arts <Accrue>
**0.80 "ballet"
**0.50 "drama"
**0.50 'dance'
**0.80 "opera"
**0.80 "symphony"
**0.90 "chamber music"
**"Isaac Stern"
*film <Accrue>
**directors <Filter>
/definition="title CONTAINS Truffaut"
*visual-arts <Accrue>
literature <Accrue>
philosophy <Accrue>
language <Accrue>
history <Accrue>
$$
```

トピックのアウトライン・ファイルはテキスト・エディタで作成できます。

トピックの提供

ユーザに提供するトピックは、mktopics ユーティリティによって生成されたトピック・セットに含まれていなければなりません。mktopics によって生成された Verity のトピック・セットは、すべての Verity アプリケーションで使用できます。1 つのトピック・セットは最大 20,000 個のトピック定義をサポートします。1 つのトピック・セットに許可されたトピックの正確な数は、トピックを定義するために使用する Verity クエリ言語によって異なります。

設定プロセス

トピックをユーザに提供するには、次のような 3 段階のプロセスがあります。

1　トピックのアウトライン・ファイルを使用してトピック定義を作成する。

2　トピック・セットを生成する。トピック・セットは mktopics ユーティリティを使用して作成できます。mktopics ユーティリティは、トピック・セットを作成し、特定のコレクションに関するトピックのインデックスも作成します。

3　トピック・セットを拡張型全文検索エンジンにインポートする。

トピックの知識ベース

この項では、知識ベースの基本的な機能と、知識ベースのトピックを定義するために使用する編成フォーマットについて説明します。

ここでは、トピックの知識ベースの次の面について説明します。

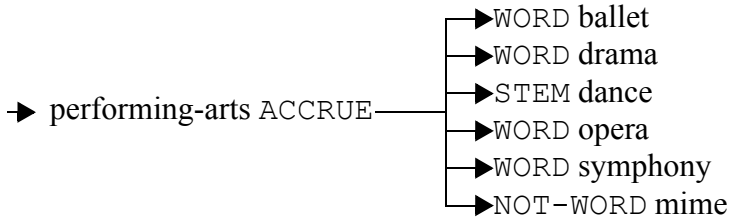
- トピックを組み合わせて知識ベースを構成する
- トピックの構造
- トピックとサブトピックの関係
- トピックのタイプ
- トピックの命名

トピックを組み合わせて知識ベースを構成する

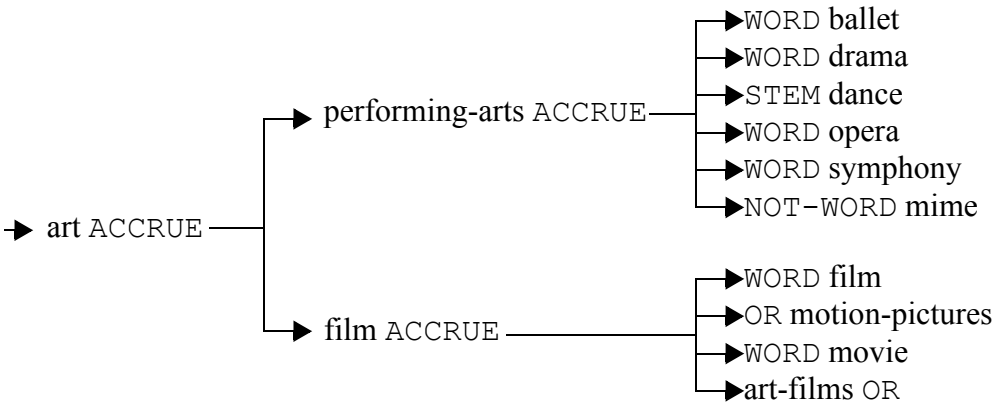
トピックとは、概念、またはサブジェクト領域に関連する情報を単にグループ化したものをいいます。そして、このようなトピックと呼ばれる概念をグループ化したものが知識ベースです。トピックを知識ベース化することで、ユーザはトピックとして保存された概念を手軽に調べることができます。

トピックのサブジェクト領域は一般に、そのトピックが持つ名前で識別されます。下の例では、トピックのサブジェクトは *performing-arts* です。このトピックは、**名前** (*performing-arts*) と**形跡**トピック (*ballet*、*musical*、*dance*、*opera*、*symphony*、*drama*) の 2 つの構成要素から構成されています。

演算子と**変更子**は、関連する形跡トピックを結び付ける働きをします。演算子は、形跡トピックに適用される論理を表します。この論理は、検索するドキュメントの種類の条件を定義します。変更子は、形跡トピックにより詳細な論理を適用します。たとえば、変更子は、ある形跡トピックを含むドキュメントが結果のリストに含まれないように指定できます。



トピックの構造は、トピックが追加されるにつれ、より高度になります。次の例では、film トピックが構造に追加され、最上位トピックが art になっています。この構造では、performing-arts と film は art トピックのサブトピックです。



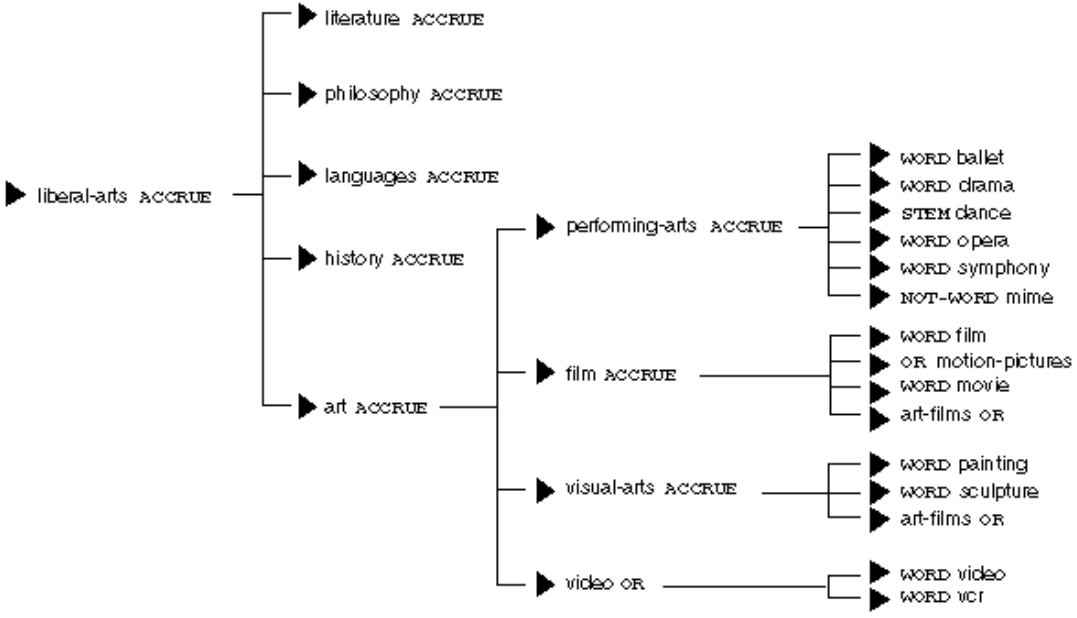
高度なトピックは、最上位トピック、サブトピック、形跡トピックで構成されます。これらの要素は、トピックに関連するサブジェクト領域を決定します。一般的に、知識ベースはいくつかの最上位トピックから成ります。サブトピックと形跡トピックは複数の最上位トピックで使用できます。

トピックの構造

トピックの構造は、トピックが検索処理でどのように解釈されるかに影響します。概念を正確に表すようにトピックを設計するには、次に説明するコンポーネントを使用してトピック構造を定義する必要があります。

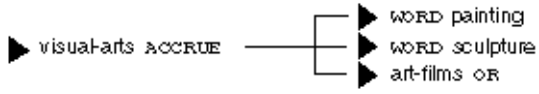
最上位トピック

最上位トピックは、トピック構造で一番上に定義されているトピックです。最上位トピックは、Verity 検索エージェントで検索するサブジェクト領域を表します。次の例の literature、philosophy、languages、history、art は、最上位トピックの liberal-arts を構成する最上位サブトピックと考えることができます。



サブトピック

サブトピックは、最上位トピックと形跡トピックの間のレベルを形成します。サブトピックの名前は、その下のサブトピックまたは形跡トピックが全体として示すサブジェクト領域を表します。たとえば次に示す visual-arts サブトピックには、関連するいくつかのワード、または形跡トピックが含まれます。

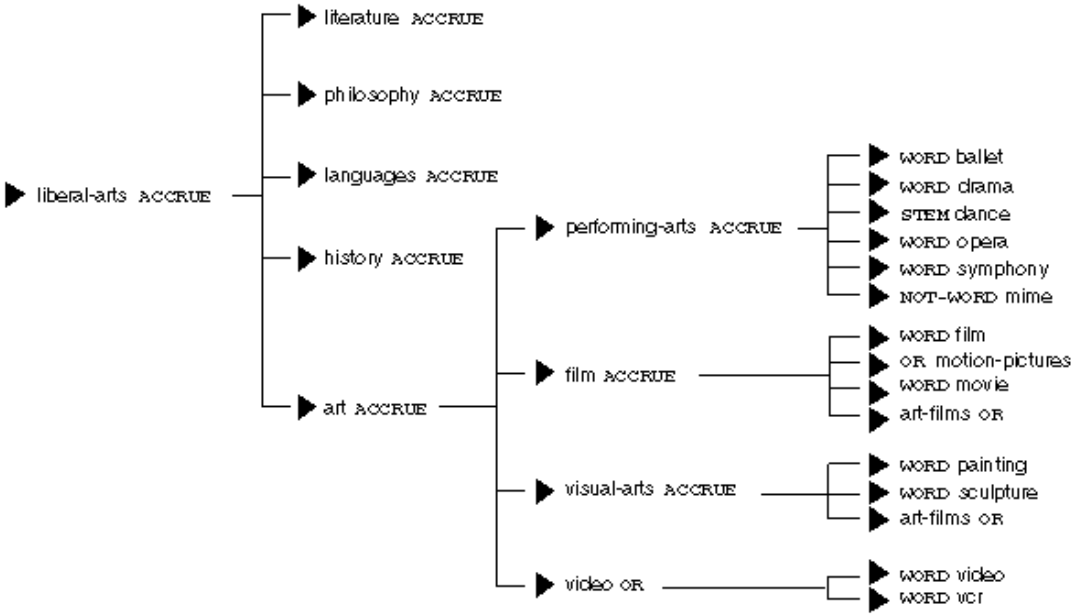


形跡トピック

形跡トピックは、トピック構造の最小単位です。形跡トピックは、英数字の組み合わせで構成された文字列です。この英数字は最大で 128 文字です。

トピックとサブトピックの関係

各トピックとその関連サブトピックは、階層的な親と子の関係を形成します。次の例では、サブトピックの performing-arts、film、visual-arts、video は art トピックの子です。art トピック自体は、liberal-arts トピックの子です。liberal-arts トピックも、構造内で連続した上位の親トピックの子になっている場合があります。



トピックを使用して検索を実行するとき、そのトピックによって定義されたサブジェクト領域には、そのサブトピック、サブトピックのサブトピックというように、その構造の形跡トピックまでが連続して含まれます。検索に使用するトピックの直接の子でないトピックは、検索に含まれません。

たとえば上の例では、film トピックを使用する検索で、Verity 検索エンジンは、film、motion pictures、movies、art films についての情報が入っているドキュメントを検索します。この例では、トピックの performing-arts、visual-arts、または video に関連するドキュメントは検索されません。これらのトピックは film トピックの子ではないからです。しかし art トピックを使用すると、art トピックのすべての子に関連するドキュメントが検索され、これには performing-arts、film、visual-arts、video も含まれます。

トピックの最大数

知識ベースを表す単一のトピック・セットは、最大 20,000 個のトピックで構成することができます。これは、最上位トピック、サブトピック、形跡トピックを合わせた数字です。ただし 1,000 個のサブトピックを含むトピックを検索で使用すると、メモリ容量の制限を超える場合があります。

トピックの命名について

- トピックの名前の長さは、ハイフンとアンダースコアを含めた英数字で最大 128 文字です。
- トピック名と形跡トピックは通常、大文字と小文字を区別しません。形跡トピックは、すべて大文字 (APPLE)、語頭のみ大文字 (Apple)、すべて小文字 (apple) のいずれでも表すことができます。検索を実行するときは大文字と小文字は区別されません。したがって、形跡トピックに APPLE と入力された場合、Verity 検索エンジンは、“APPLE”、“Apple”、“apple” を含むドキュメントを選択します。

  ただし、CASE 変更子を使用して、形跡トピックに入力された大文字と小文字を区別するように指定することもできます。

Verity クエリ言語

この項では Verity クエリ言語について説明します。この言語は演算子と変更子で構成され、トピックの作成に使用します。演算子は、トピック作成のために結合できる検索要素に適用する論理を表します。この論理は、検索するドキュメントの種類の条件を定義します。変更子は、より詳細な論理を検索要素に適用します。たとえば変更子を使用して、検索要素の大文字と小文字を区別するように指定できます。

この項で説明する情報は、次のとおりです。

- クエリ言語の概要
- 演算子の優先度の規則
- トピックのアウトラインの例
- 演算子の参照
- 変更子の参照

クエリ言語の概要

Verity クエリ言語は演算子と変更子で構成されます。演算子と変更子はいずれも、検索要素に適用される論理を表します。この論理は、ドキュメントを取り出す際に満たさなくてはならない条件を定義します。演算子は次のようにタイプ別に分けることができます。

- 形跡演算子
- 近接演算子
- 関係演算子
- 概念演算子
- ブール演算子

変更子は、演算子によって適用される論理を拡張するもので、演算子と組み合わせて使用します。

形跡演算子

形跡演算子は、1 つの検索ワードを複数の関連ワードのリストに拡張し、この関連ワードも検索します。形跡演算子を使用して検索する場合、拡張されたワード・リスト内のワードのオカレンスが 1 つ以上含まれるドキュメントは、指定されたワードの他にその同義語も含むドキュメントです。形跡演算子を使用して取り出されるドキュメントは、MANY 変更子を使用しない限りは関連性のランク付けがされていません。詳細については、「MANY 変更子」(106 ページ) を参照してください。表 8-1 は、個々の形跡演算子を示します。

表 *8-1:* 形跡演算子

演算子名	説明
WORD	指定したワードのインスタンスを 1 つ以上含むドキュメントを選択します。
STEM	指定した検索ワードの語尾変化したものを 1 つ以上含むドキュメントを選択します。
THESAURUS	指定したワードの同義語を 1 つ以上含むドキュメントを選択します。
WILDCARD	変数を含む文字列に一致する文字列を含むドキュメントを選択します。
SOUNDEX	指定したワードと「音が似ている」、または文字のパターンが似ているワードを 1 つ以上含むドキュメントを選択します。
NEAR/N	検索範囲を拡大し、入力したワードに加え、検索対象ワードに類似したワードを含めます。この演算子は類似ワードを識別する「近接パターン一致」を実行します。

近接演算子

近接演算子は、ドキュメント内の特定のワードの相対的なロケーションを指定します。つまり、指定されたワードが取り出されるドキュメント内で同じフレーズ、同じ段落、または同じ文の中になくてはなりません。NEAR と NEAR/N 演算子の場合、取り出されるドキュメントは指定されたワードとの近似に基づいて関連性をランク付けされます。近接演算子をネストすると、最も広いスコープを持つものが最初に使用されます。つまり、フレーズまたは個々のワードが SENTENCE 演算子または PARAGRAPH 演算子内に表示され、SENTENCE 演算子は PARAGRAPH 演算子内に表示されます。表 8-2 に、個々の近接演算子を示します。

表 *8-2:* 近接演算子

演算子名	説明
IN	指定した値が 1 つ以上のドキュメント･ゾーンに含まれるドキュメントを選択します。ドキュメント・ゾーンは、ドキュメントの要約、日付、本文テキストなど、ドキュメントの領域を表します。
PHRASE	指定したフレーズを含むドキュメントを選択します。フレーズは、特定の順序で並ぶ 2 つ以上のワードのグループです。
SENTENCE	指定したすべてのワードが 1 つの文に含まれるドキュメントを選択します。
PARAGRAPH	指定したすべての検索要素が 1 つの段落に含まれるドキュメントを選択します。
NEAR	指定した検索ワードが互いに近くにあるドキュメントを選択します。
NEAR/N	2 つ以上の検索ワードが互いに N ワード以内に存在するドキュメントを選択します。N は整数です。

関係演算子

関係演算子は、コレクション内で定義されているドキュメント・フィールド (AUTHOR など) を検索します。この演算子は、指定したフィールド値を含むドキュメントを選択して、フィルタリングを実行します。関係演算子とともに使用するフィールドには英数字を指定できます。関係演算子を使用して取り出されるドキュメントは、関連性によってランク付けされません。また、MANY 変更子は関係演算子と併用できません。

トピックを作成するときは必ず、関係演算子とともに FILTER 特別演算子を使用します。適切な構文については、この項の「トピックのアウトラインの例」(後掲) の “visual-arts” トピックの下の例を参照してください。

= (等しい)、> (より大きい)、>= (より大きいかまたは等しい)、< (より小さい)、<= (より小さいかまたは等しい) などの関係演算子を、数値と日付の比較に使用できます。

テキストの比較に使用できる関係演算子は、次のとおりです。

表 *8-3:* 関係演算子

演算子名	説明
CONTAINS	指定したワードまたはフレーズと、特定のドキュメント・フィールドに格納された値とが一致するドキュメントを選択します。
MATCHES	指定した文字列と、特定のドキュメント・フィールドに格納された値とが一致するドキュメントを選択します。
STARTS	指定した文字列と、特定のドキュメント・フィールドに格納された値の開始文字とが一致するドキュメントを選択します。
ENDS	指定した文字列と、特定のドキュメント・フィールドに格納された値の終了文字とが一致するドキュメントを選択します。
SUBSTRING	指定した文字列と、特定のドキュメント・フィールドに格納された値の文字列の一部が一致するドキュメントを選択します。

概念演算子

概念演算子は、検索要素の意味を組み合わせて、ドキュメント内の概念を識別します。概念演算子を使って取り出されたドキュメントは関連性によってランク付けされます。表 8-4 に、個々の概念演算子を示します。

表 *8-4:* 概念演算子

演算子名	説明
AND	指定したすべての検索要素を含むドキュメントを選択します。
OR	1 つ以上の検索要素の形跡を示すドキュメントを選択します。
ACCRUE	指定した検索要素を 1 つ以上含むドキュメントを選択します。

ブール演算子

ブール演算子をトピックに割り当てて、そのトピックのいずれかまたはすべての子を含むドキュメントを取り出すことができます。概念演算子を使用して作成したトピックと異なり、ブール演算子はウェイトを許可しません。表 8-5 に、個々のブール演算子を示します。

表 *8-5:* ブール演算子

演算子名	説明
ALL	トピックのすべての子を含むドキュメントを選択します。
ANY	トピックの子を 1 つ以上含むドキュメントを選択します。

変更子

変更子は演算子の動作に影響します。表 8-6 に、個々の変更子を示します。

表 *8-6:* 変更子

演算子名	説明
CASE	大文字と小文字を区別して検索します。
MANY	ドキュメント内のワードまたはフレーズの密度をカウントして、取り出されたドキュメントに対し、関連性によってランク付けされたスコアを生成します。
NOT	指定したワードまたはフレーズの形跡を示すドキュメントを除外します。
ORDER	検索要素の発生順を指定します。

演算子の優先度の規則

Verity 検索エンジンは、優先度の規則を使用して演算子の割り当て方法を決定します。この規則では、トピックへの割り当ての際に一部の演算子は他の演算子よりランクが高くなります。また、この規則はドキュメントの選択方法に影響します。

表 8-7 は、優先度の規則が演算子にどのように適用されるかを示します。

表 *8-7:* 優先度の規則

演算子	優先度	優先度の決定方法
AND OR ACCRUE	最も高い優先度	概念演算子は、他の演算子よりも優先度が高い。したがって、概念演算子を使用するトピックのサブトピックには、下の「インクリメンタルな優先度」または「最も低い優先度」にリストされたどの演算子も割り当てることができる。
ALL PARAGRAPH SENTENCE NEAR NEAR/N PHRASE ANY	インクリメンタルな優先度 (降順)	近接演算子は、ドキュメント内に存在するインクリメンタルな範囲を示す。近接演算子を使用するトピックのサブトピックには、この演算子より優先度の順位が 1 つ下の演算子を割り当てることができる。したがって、フレーズはワードより優先され、文はフレーズまたはワードより優先される。段落は、文、フレーズ、ワードより優先される。
WORD STEM SOUNDEX WILDCARD THESAURUS	最も低い優先度	形跡演算子は、トピック構造の最下位に属する。形跡演算子はドキュメント内に含まれるワードとともに使用されるため、すべての形跡演算子の優先度が同じになる。

優先度違反を避けるため、ANY と ALL は、子トピックに概念演算子 (AND、OR、ACCRUE) が含まれる親トピックでは使用しないでください。ANY または ALL を使用するトピックには、変数ウェイトを割り当てることはできません。したがって、これらの演算子は、変数ウェイト (AND、OR、ACCRUE など) が許可される子トピックのある親トピックでは使用できません。ANY と ALL を使用するトピックでは、評価が「存在する」または「存在しない」(スコアが 0.00 または 1.00) に制限されます。このようなトピックの子は、基準を満たすとスコアが自動的に 1.00 になり、基準を満たさないと自動的に 0.00 になります。つまり、このような子に 0.80 などの変数ウェイトを割り当てても意味がありません。

トピックのアウトラインの例

以下に、トピックのアウトライン・ファイルで作成するトピックの例を示します。

```
$Control:1
art <Accrue>
*performing-arts <Or>
**0.80 "drama"
**0.50 "theater"
**0.80 'dance'
*film <And>
**0.90 "cinema"
**0.90 "documentary"
**newsreel <Filter>
/definition="DATE  >= 05/01/96"
*film-makers <Accrue>
**"Woody Allen"
*film-making <Paragraph>
**"direct"
**"produce"
*visual-arts <Accrue>
**sculpture <In>
/zonespec="title"
**painters <Filter>
/definition="Title MATCHES Famous Painters"
**<Thesaurus>
/wordtext="paint"
literature <Accrue>
*novels <Near>
**0.80 "Proust"
**0.80 "Remembrance" <Case>
*american-novel <Sentence>
**"American"
**"novel"
history <Accrue>
*<Wildcard>
```

```
/wordtext="histor*"
music <Accrue>
*jazz
**"bebop"
**<Not> "fusion"
*classical
**"Italian opera"
$$
```

演算子の参照

各演算子を以下にアルファベット順で示します。ここに示す演算子の多くは、前項にあるトピックのアウトラインで例を見ることができます。

ACCRUE 演算子

指定した検索要素を 1 つ以上含むドキュメントを選択します。有効な検索要素は 2 つ以上のワードまたはフレーズです。選択されたドキュメントは関連性によってランク付けされます。

ACCRUE 演算子は、選択されたドキュメントに、そのドキュメント内の各検索要素の存在に対応して「多いほどよい」方式でスコアを付けます。つまり、ドキュメント内で検索要素が多く見つかるほど、そのドキュメントのスコアは高くなります。「トピックのアウトラインの例」(96 ページ) にあるアウトライン・ファイルのサンプルに、ACCRUE 演算子の例がいくつか示されています。

ALL 演算子

指定したすべての検索要素を含むドキュメントを選択します。ACCRUE 演算子と異なり、ALL 演算子を使用するときはウェイトを割り当てることができません。

AND 演算子

指定したすべての検索要素を含むドキュメントを選択します。AND 演算子を使用して選択されたドキュメントは関連性によってランク付けされます。「トピックのアウトラインの例」は、AND 演算子を “film” トピックとともに使用する方法を示しています。この例では、検索ワードとともに、05/01/96 より大きいかまたは等しい日付を含むドキュメントだけが選択され、スコアに従ってランク付けされています。

ANY 演算子

指定した検索要素を 1 つ以上含むドキュメントを選択します。ACCRUE 演算子と異なり、ANY 演算子を使用するときはウェイトを割り当てることができません。

CONTAINS 演算子

指定したワードまたはフレーズと、特定のドキュメント・フィールドに格納された値とが一致するドキュメントを選択します。CONTAINS 演算子を使用するときは、検索するフィールドの名前と、検索対象のワードまたはフレーズを指定します。

CONTAINS 演算子を指定すると、ドキュメント・フィールドに格納されたワードは連続した個別の単位として解釈されます。検索基準として、これらの単位を 1 つ以上指定できます。複数のワードを指定する場合、各ワードは連続して隣り合っている必要があり、さらにブランク・スペースで区切ります。CONTAINS は FILTER 演算子とともに使用します。

CONTAINS の構文は MATCHES の構文と同じです。「トピックのアウトラインの例」(96 ページ) の “visual arts” トピックの下にある MATCHES の例を参照してください。この例では、TITLE フィールドがコレクションのために作成されていることを前提にしています。

CONTAINS 演算子は非英数字を認識しません。CONTAINS 演算子は非英数字をスペースと解釈し、区切られた値を別個の単位として処理します。たとえばスラッシュ (/) を有効な文字として定義した場合、OS/2 のようにスラッシュを含む検索基準を入力すると、“OS” と “2” は別個の単位として処理されます。

CONTAINS 演算子は、どの文字がワードに含まれているかという定義について *style.lex* ファイルを参照しません。

ENDS 演算子

指定した文字列と一致するドキュメントを選択します。ENDS は FILTER 演算子とともに使用します。ENDS の構文は MATCHES の構文と同じです。「トピックのアウトラインの例」(96 ページ) の “visual arts” トピックの下にある MATCHES の例を参照してください。この例では、TITLE フィールドがコレクションのために作成されていることを前提にしています。

= (EQUALS) 演算子

ドキュメント・フィールドの値が、指定した検索文字列と完全に一致するドキュメントを選択します。EQUALS は FILTER 演算子とともに使用します。EQUALS の構文は GREATER THAN OR EQUAL TO の構文と同じです。「トピックのアウトラインの例」(96 ページ) の “film” トピックの下にある GREATER THAN OR EQUAL TO の例を参照してください。この例では、DATE フィールドがコレクションのために作成されていることを前提にしています。

FILTER 演算子

特別な FILTER 演算子は、フィールド検索を行う関係演算子とともに使用します。適切な構文については、「トピックのアウトラインの例」(96 ページ) の “visual-arts” トピックの下にある例を参照してください。

> (GREATER THAN) 演算子

ドキュメント・フィールドの値が、指定した検索文字列より大きいドキュメントを選択します。GREATER THAN は FILTER 演算子とともに使用します。GREATER THAN の構文は GREATER THAN OR EQUAL TO の構文と同じです。「トピックのアウトラインの例」(96 ページ) の “film” トピックの下にある GREATER THAN OR EQUAL TO の例を参照してください。この例では、DATE フィールドがコレクションのために作成されていることを前提にしています。

>= (GREATER THAN OR EQUAL TO) 演算子

ドキュメント・フィールドの値が、指定した検索文字列より大きいかまたは等しいドキュメントを選択します。GREATER THAN OR EQUAL TO は FILTER 演算子とともに使用します。「トピックのアウトラインの例」(96 ページ) の “film” トピックの下にある例を参照してください。この例では、DATE フィールドがコレクションのために作成されていることを前提にしています。

< (LESS THAN) 演算子

ドキュメント・フィールドの値が、指定した検索文字列より小さいドキュメントを選択します。LESS THAN は FILTER 演算子とともに使用します。LESS THAN の構文は GREATER THAN OR EQUAL TO の構文と同じです。「トピックのアウトラインの例」(96 ページ) の “film” トピックの下にある GREATER THAN OR EQUAL TO の例を参照してください。この例では、DATE フィールドがコレクションのために作成されていることを前提にしています。

<= (LESS THAN OR EQUAL TO) 演算子

ドキュメント・フィールドの値が、指定した検索文字列より小さいかまたは等しいドキュメントを選択します。LESS THAN OR EQUAL TO は FILTER 演算子とともに使用します。LESS THAN OR EQUAL TO の構文は GREATER THAN OR EQUAL TO の構文と同じです。「トピックのアウトラインの例」(96 ページ) の “film” トピックの下にある GREATER THAN OR EQUAL TO の例を参照してください。この例では、DATE フィールドがコレクションのために作成されていることを前提にしています。

IN 演算子

指定した値が 1 つ以上のドキュメント・ゾーンに含まれるドキュメントを選択します。ドキュメント・ゾーンは、ドキュメントの要約、日付、本文テキストなど、ドキュメントの領域を表します。IN 演算子は、ドキュメント・ゾーンがコレクション内で定義されている場合のみ有効です。IN 演算子を使用してゾーンが定義されていないコレクションを検索すると、ドキュメントは選択されません。さらに、指定したゾーン名がコレクションで定義されたゾーン名と一致している必要があります。特定のコレクションに対してどのゾーンが定義されているかについては、コレクション管理者にお問い合わせください。「トピックのアウトラインの例」(96 ページ) は、IN をワード “sculpture” と TITLE ゾーンとともに使用する方法を示しています。

MATCHES 演算子

指定した文字列と、特定のドキュメント・フィールドに格納された値とが一致するドキュメントを選択します。MATCHES 演算子を使用するときは、検索を行うフィールドの名前と、検索対象のワード、フレーズ、または数値を指定します。

CONTAINS 演算子と異なり、MATCHES 演算子とともに指定する検索基準は、選択されるドキュメントのフィールド値と完全に一致する必要があります。MATCHES 演算子を指定すると、値の一部だけが一致する検索文字列のオカレンスは選択されず、検索文字列全体が完全に一致する値だけが選択されます。

疑問符 (?) を使用して文字列内の個々の変数文字を表したり、アスタリスク (*) を使用して文字列内の複数の変数文字に一致させたりできます。

MATCHES は FILTER 演算子とともに使用します。「トピックのアウトラインの例」(96 ページ) は、MATCHES をフレーズ “famous painters” と TITLE フィールドとともに使用する方法を示しています。この例では、TITLE フィールドがコレクションのために作成されていることを前提にしています。

NEAR 演算子

指定した検索ワードが互いに近くにあるドキュメントを選択します。ドキュメントのスコアは、検索ワード間の相対的なワード数に基づいて計算されます。たとえば検索式に 2 つのワードがあり、これらのワードがドキュメント内で互いに連続している場合 (領域サイズが 2 ワード分の長さになる)、このドキュメントに割り当てられるスコアは 1.00 になります。このように、すべての検索ワードが最も小さい領域内に含まれるドキュメントが、常に最も高いスコアになります。検索ワードが互いに 1000 ワード以内にないドキュメントは選択されません。この場合には、検索ワードがこのドキュメントのコンテキストの中で関連した意味を持つには離れすぎていると考えられるからです。

NEAR 演算子は、入力する検索ワードが相互に近接していなければならないという点で、他の近接演算子と同じです。ただし、他の近接演算子と異なり、NEAR 演算子は相対的な近さを計算し、その計算に基づいてスコアを割り当てます。

「トピックのアウトラインの例」(96 ページ) は、NEAR 演算子を “novels” トピックとともに使用する方法を示しています。

NEAR/N 演算子

2 つ以上の検索ワードが互いに N ワード以内に存在するドキュメントを選択します。N は整数です。ドキュメントのスコアは、指定したワードが N ワード以下しか離れていない場合、その相対距離に基づいて計算されます。指定したワードが N ワードよりも離れているドキュメントは選択されません。たとえば、検索式 NEAR/5 を使用して、互いに 5 ワード以内にある 2 つのワードを検索する場合、互いに 3 ワード以内に指定したワードがあるドキュメントのスコアは、互いに 5 ワード以内に指定したワードがあるドキュメントより高くなります。

N 変数には、1 から 1,024 までの整数を指定できます。NEAR/1 と指定すると、互いに隣接している 2 つのワードが検索されます。N が 1,000 以上の場合は、NEAR/1000 のようにカンマを使わずに値を指定してください。

NEAR/N 演算子は、入力する検索ワードが相互に近接していなければならないという点で、他の近接演算子と同じです。ただし、他の近接演算子と異なり、NEAR/N 演算子は相対的な近さに基づいてスコアを割り当てます。

OR 演算子

1 つ以上の検索要素の形跡を示すドキュメントを選択します。OR 演算子を使用して選択されたドキュメントは関連性によってランク付けされます。「トピックのアウトラインの例」(96 ページ) は、OR 演算子を “performing-arts” トピックとともに使用する方法を示しています。

PARAGRAPH 演算子

指定したすべての検索要素が 1 つの段落に含まれるドキュメントを選択します。有効な検索要素は 2 つ以上のワードまたはフレーズです。検索要素は、連続した順序でもランダムな順序でも指定できます。ドキュメントは、検索要素が同じ段落内に出現するかぎり取り出されます。「トピックのアウトラインの例」(96 ページ) は、PARAGRAPH 演算子を “film-making” トピックとともに使用する方法を示しています。

PHRASE 演算子

指定したフレーズを含むドキュメントを選択します。フレーズは、特定の順序で並ぶ 2 つ以上のワードのグループです。PHRASE 演算子は、形跡フィールドに複数のワードを入力するときに使用する必要があります。PHRASE 演算子を指定したワードは二重引用符に囲まれて表示されます。「トピックのアウトラインの例」(96 ページ) は、PHRASE 演算子を “Woody Allen” および “Italian opera” とともに使用する方法を示しています。

SENTENCE 演算子

指定したすべてのワードが 1 つの文に含まれるドキュメントを選択します。検索要素は、連続した順序でもランダムな順序でも指定できます。ドキュメントは、検索要素が同じ 1 文内に出現するかぎり取り出されます。「トピックのアウトラインの例」(96 ページ) は、SENTENCE 演算子を “american-novel” トピックとともに使用する方法を示しています。

SOUNDEX 演算子

指定したワードと「音が似ている」、または文字のパターンが似ているワードを 1 つ以上含むドキュメントを選択します。指定したワードと同じ文字で始まるワードが選択されます。たとえば、SOUNDEX を “sale” とともに使用すると、選択されるドキュメントには “sale”、“sell”、“seal”、“shell”、“soul”、“scale” などのワードが含まれます。MANY 変更子を指定しない限り、ドキュメントは関連性によってランク付けされません。

STARTS 演算子

指定した文字列と、特定のドキュメント・フィールドに格納された値の開始文字とが一致するドキュメントを選択します。STARTS は FILTER 演算子とともに使用します。STARTS の構文は MATCHES の構文と同じです。「トピックのアウトラインの例」(96 ページ) の “visual arts” トピックの下にある MATCHES の例を参照してください。この例では、TITLE フィールドがコレクションのために作成されていることを前提にしています。

STEM 演算子

指定した検索ワードの語尾変化したものを 1 つ以上含むドキュメントを選択します。STEM 演算子を指定したワードは一重引用符に囲まれて表示されます。「トピックのアウトラインの例」では、“dance” というワードが STEM 演算子とともに使用されています。したがって選択されるドキュメントには、“dance” のほかに “dances”、“danced”、“dancing” などのワードが含まれます。

SUBSTRING 演算子

指定した文字列と、特定のドキュメント・フィールドに格納された値の文字列の一部が一致するドキュメントを選択します。文字列を構成する文字の出現箇所は、フィールド値の最初、中間、または最後のいずれでもかまいません。SUBSTRING の構文は MATCHES の構文と同じです。「トピックのアウトラインの例」の “visual arts” トピックの下にある MATCHES の例を参照してください。この例では、TITLE フィールドがコレクションのために作成されていることを前提にしています。

THESAURUS 演算子

指定したワードの同義語を 1 つ以上含むドキュメントを選択します。たとえばワード “altitude” を THESAURUS 演算子とともに使用する場合、選択されるドキュメントには “height” や “elevation” などのワードが含まれます。MANY 変更子を指定しない限り、ドキュメントは関連性によってランク付けされません。

TYPO/N 演算子

TYPO/N 演算子は、指定したワードとそれに類似したワードが含まれるように検索を拡大します。オプションの *N* 変数は、検索対象ワードと突き合わせたワードの間の最大誤り数を指定します。たとえば、TYPO mouse は、“house”、“louse”、“moose” などのワードを含むドキュメントを返します。

WILDCARD 演算子

変数を含む文字列に一致する文字列を含むドキュメントを選択します。WILDCARD 演算子を使用すると、変数のある検索文字列を定義できます。これを使用してドキュメント内の関連ワードの一致を特定します。「トピックのアウトラインの例」(96 ページ) は、文字列 “histor*” を使用して “history”、“historical”、“historian” などのワードを検索する方法を示しています。MANY 変更子を指定しない限り、ドキュメントは関連性によってランク付けされません。

ワイルドカード特殊文字の使用

次のワイルドカード文字は、WILDCARD 演算子を指定する検索文字列の変数部分を表すために使用できます。

表 *8-8:* ワイルドカード特殊文字

文字 (C)	機能
?	英数字の 1 つとして指定する。?an とすると、“ran”、“pan”、“can”、“ban” が検索される。疑問符を使用するときは、WILDCARD 演算子を指定する必要はない。疑問符は、セット内 ([ ]) または代替パターン内 ({ }) では無視される。
*	0 個以上の英数字として指定する。corp* とすると、“corporate”、“corporation”、“corporal”、“corpulent” が検索される。アスタリスクを使用するときは、WILDCARD 演算子を指定する必要はない。また、アスタリスクはワイルドカード文字列の最初の文字としては使用しない。アスタリスクはセット内 ([ ]) または代替パターン内 ({ }) では無視される。
[ ]	セット内の任意の 1 文字を指定する。<WILDCARD> ‘c[auo]t‘ とすると、“cat”、“cut”、“cot” が検索される。セットを含むワードはバック引用符 (`) で囲み、セット内にはスペースを入れない。
{ }	カンマで区切られた各パターンの 1 つを指定する。<WILDCARD> ‘bank{s,er,ing}‘ とすると、“banks”、“banker”、“banking” が検索される。パターンを含むワードはバック引用符 (`) で囲み、パターン内にはスペースを入れない。
^	セットに含まれていない任意の 1 文字を指定する。<WILDCARD> ‘st[^oa]ck‘ とすると、“stock” と “stack” は除外されるが “stick” と “stuck” は検索される。脱字記号 (^) は、セットを開始する左角カッコ ([) の後の最初の文字とする。
-	セット内の文字の範囲を指定する。<WILDCARD> ‘c[a-r]t‘ とすると、“cat” から “crt” までのすべての 3 文字のワードが検索される。

非英数字の検索

非英数字を検索できるのは、検索するコレクションの作成に使用した *style.lex* ファイルが、検索する非英数字を認識するように設定されている場合だけに限られます。詳細については、コレクション管理者にお問い合わせください。

ワイルドカード文字をリテラルとして検索する

上の表に示したワイルドカード文字は、円記号 (¥) で区切られていない限り、ワイルドカード文字として解釈され、リテラル文字としては解釈されません。ワイルドカード文字がワイルドカード文字列内でリテラルとして解釈されるようにするには、その文字の前に円記号を付ける必要があります。たとえば、ワイルドカード文字列内でリテラル・アスタリスク (*) に相当させるには、その文字を次のように区切ります。

```
<WILDCARD> a¥*
```

特殊文字をリテラルとして検索する

次の非英数字は、特殊な内部機能を実行し、デフォルトではワイルドカード文字列内でリテラルとして処理されません。

- カンマ ,
- 左右のカッコ ( )
- 二重引用符 “
- 円記号 ¥
- アットマーク @
- 左中カッコ {
- 左角カッコ [
- 左山カッコ <
- バック引用符 `

特殊文字をリテラルとして解釈するには、ワイルドカード文字列全体をバック引用符 (`) で囲みます。たとえば、ワイルドカード文字列 “a{b” を検索するには、この文字列を次のようにバック引用符で囲みます。

```
<WILDCARD> `a{b`
```

リテラルのバック引用符文字 (`) を含むワイルドカード文字列を検索するには、次のように 2 つのバック引用符を使用して、ワイルドカード文字列全体をバック引用符 (`) で囲みます。

```
<WILDCARD> `*n``t`
```

バック引用符を検索できるのは、検索するコレクションの作成に使用した *style.lex* ファイルが、バック引用符文字を認識するように設定されている場合だけに限られます。詳細については、コレクション管理者にお問い合わせください。

WORD 演算子

指定したワードのインスタンスを 1 つ以上含むドキュメントを選択します。WORD 演算子を指定したワードは二重引用符に囲まれて表示されます。「トピックのアウトラインの例」(96 ページ) には、WORD 演算子の多くのインスタンスが示されています。

変更子の参照

変更子を使用すると、演算子の動作を詳細に指定できます。たとえば、ある演算子とともに CASE 変更子を使用すると、入力した検索ワードの大文字と小文字の区別も検索要素として指定できます。変更子には CASE、MANY、NOT、ORDER があります。次にこれらについて説明します。

CASE 変更子

CASE 変更子を、演算子の WORD または WILDCARD とともに使用すると、指定したワードまたはフレーズの大文字と小文字を区別した検索が行われます。

デフォルトでは、検索ワードまたはフレーズのオカレンスを含むドキュメントが、大文字と小文字の区別なしに取り出されます。CASE 変更子を使用するには、検索ワードまたはフレーズを、取り出されたドキュメントで表示させたい形式で入力するだけです。つまりすべて大文字か、大文字と小文字の混合か、またはすべて小文字で入力します。「トピックのアウトラインの例」(96 ページ) は、ワード “Remembrance” を CASE 変更子とともに使用して、プルーストの小説の題名『Remembrance of Things Past (失われた時を求めて)』の最初の語を参照する方法を示しています。

MANY 変更子

ドキュメント内のワード、語尾変化したさまざまな語、またはフレーズの密度をカウントして、取り出されるドキュメントの関連性によってランク付けされたスコアを生成します。ドキュメント・テキストの量に比例して、ワード、語尾変化語、またはフレーズのオカレンスが多いほど、そのドキュメントが取り出されたときのスコアは高くなります。MANY 変更子はドキュメント・テキストに比例した密度をカウントするため、あるワードのオカレンスをより多く含む長いドキュメントよりも、オカレンスが少なくても短いドキュメントの方がスコアが高くなる場合があります。

MANY 変更子とともに使用できる演算子は、WORD、WILDCARD、STEM、SOUNDEX、PHRASE、SENTENCE、PARAGRAPH、THESAURUS です。

MANY 変更子は、AND、OR、ACCRUE、または関係演算子とともに使用することはできません。

NOT 変更子

NOT 変更子は、ワードまたはフレーズとともに使用して、そのワードまたはフレーズの形跡を示すドキュメントを除外します。「トピックのアウトラインの例」(96 ページ) は、NOT 変更子を使用して、“bebop” は含むが “fusion” は含まないドキュメントを取り出す方法を示しています。

ORDER 変更子

ORDER 変更子は、検索要素が出現する順序を表すのに使用します。検索値が指定した順序でドキュメントに現れない場合、そのドキュメントは選択されません。ORDER 変更子は必ず演算子の直前に置きます。

ORDER 変更子とともに使用できる演算子は、ALL、PARAGRAPH、SENTENCE、NEAR/N だけです。

ウェイトとドキュメントの重要性

このセクションでは、トピック内での検索基準へのウェイトの割り当てと、選択されたドキュメントへのウェイトの影響について説明します。ここで説明する情報は次のとおりです。

- ウェイトを許可する演算子
- ウェイトと重要性の関係
- ウェイトの割り当て
- トピックのスコア計算とドキュメントの重要性

トピックのウェイト

検索エージェントを処理する際、Verity 検索エンジンは、選択された各ドキュメントのスコアも計算します。ドキュメントのスコアは 1.0 から 0.01 までの範囲内になります。ドキュメントのスコアが高いほど、そのドキュメントが持つ関連性は強くなります。検索エージェントによって選択されたドキュメントに対するスコアの割り当てを使用して、Verity アプリケーションは、関連性によってランク付けされた結果を降順で表示します。

ドキュメントのランクは、検索基準を構成する要素によって決定されます。ドキュメントのランク付けは、検索基準にトピックが含まれているか、トピックにウェイトが含まれているかによって左右されます。

トピックを作成するときは、トピック構造にウェイトを割り当てて、そのトピック定義の特定の面が持つ相対的な重要性を示すことができます。たとえば、関連する 2 つのサブジェクトに関心があっても、そのうちの 1 つが他方より重要な場合です。トピックを構成するときには、ウェイトを割り当てなくてもかまいません。トピック・セットにインデックスが作成されるときに、必要に応じてデフォルトのウェイトが割り当てられるからです。ただし、ウェイトを割り当てることによって、検索する事項の重要性を微調整できます。

ウェイトを許可する演算子

ウェイトは、検索中に親トピックと子トピックのスコアを計算するために、演算子とともに使用します。子トピックに割り当てられるウェイトは、0.01 から 1.00 までの数です。子のウェイトは、その親に定義されているその他の子との相対的な重要性を示します。ある子のウェイトが高いほど、その子は他の子とくらべてより重要であると考えられます。

ウェイトを割り当てられるのは、次の概念演算子を使用するトピックの子に対してだけです。

- AND
- OR
- ACCRUE

近接演算子の SENTENCE と PARAGRAPH を使用するトピックにはウェイトを割り当てることはできません。これらの演算子では、子が単に存在するかしないかだけを想定します。

逆にトピックに近接演算子が割り当てられていて、そのトピックの子に AND 演算子などの概念演算子が割り当てられていた場合、その子にはウェイトを割り当てることができます。

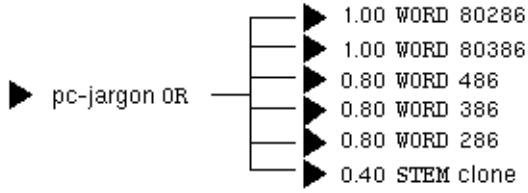
ただし、演算子がウェイトのある子を受け入れることができるからといって、トピックの子にウェイトを割り当てる必要はありません。ウェイトが割り当てられていないと、その子には演算子に基づいて自動的にウェイトが割り当てられます。演算子の AND と OR を使用するトピックの子のウェイトは 1.00、ACCRUE 演算子を使用するトピックの子のウェイトは 0.50 と想定されます。これらの演算子が変更された場合、たとえば OR 演算子が ACCRUE 演算子に変更された場合、特にウェイトを割り当てられていない子のウェイトは、演算子の変更に従って変わります。つまり、ウェイトのない AND トピックの子のウェイトが 1.00 と想定されていた場合、演算子が ACCRUE に変更されるとこの仮定のウェイトは 0.50 に変わります。

変数ウェイトを子トピックに割り当て、次にその親とともに使用する演算子を、SENTENCE などウェイト付きの子を許可しない演算子に変更したとします。Verity 検索エンジンは、ウェイトが 1.00 であると自動的に想定しますが、この演算子は有効なままです。次にこの演算子が変数ウェイト付きの子を許可する演算子に変更されると、先に割り当てた変数ウェイトが再び有効になります。

ウェイトと重要性の関係

概念演算子を使用する子トピックにウェイトを割り当てるときは、その子の相対貢献度を、トピックによって生成されるスコア全体に指定します。その子に割り当てるウェイトが大きいほど、選択されたドキュメントでその子を含むものは結果のリストで上に表示されます。このように、ウェイトは選択されたドキュメントの重要性またはランク付けに直接影響します。

たとえば、次のようなトピックがあるとします。



形跡トピック 80286 と 80386 (PC 製品で使用されるマイクロプロセッサを表す) には、1.00 のウェイトが自動的に割り当てられています。形跡トピック 486、386、286 はその親トピックについて言及している可能性が相対的に高いため、これらの形跡トピックには 0.80 のウェイトが割り当てられます。形跡トピック clone は PC クローンについて、言及している場合もあれば、まったく言及していない場合もあります。したがって、この形跡トピックには 0.40 のウェイトが割り当てられます。

このトピックとその割り当てウェイトを使用する検索エージェントは、一致したドキュメントに対して次のようなスコアを生成します。

Scores

```
1.00  01-Oct-90  New Toshiba Portable Desktop Computers Offer a Serious Alternative for Desktop Users
1.00  14-Feb-90  Technology: 'Chip Set' Unveiled for Use in Making Faster Computers
0.80  13-Feb-91  CMS Enhancements Inc. Unveils New Products
0.80  01-Oct-90  Top Selling Microsoft Windows Applications Now Available in One Convenient Package
0.80  01-Oct-90  KeyCorp Successful Bidder to Acquire New York State Divisions of Empire of America Federal
0.80  15-Feb-90  Health: U.S. Birth Control Lags
```

各形跡トピックのウェイトを変更すると、選択された結果の重要性にも影響します。この例で、形跡トピック 486 のウェイトを 0.60 に、386 を 0.45 に、286 を 0.35 に、clone を 0.20 に変更すると、選択されたドキュメントのスコアは次のように変更されます。

Scores

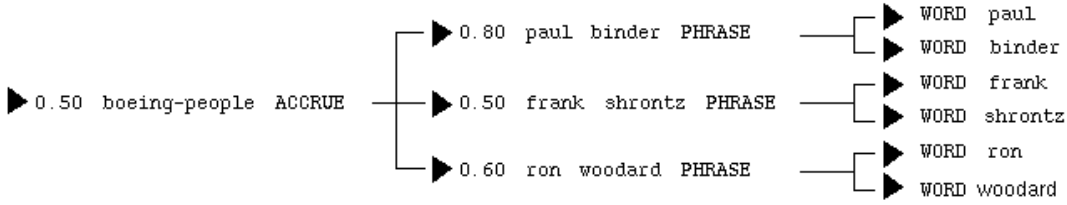
```
1.00  01-Oct-90  New Toshiba Portable Desktop Computers Offer a Serious Alternative for Desktop Users
1.00  14-Feb-90  Technology: 'Chip Set' Unveiled for Use in Making Faster Computers
0.60  13-Feb-91  CMS Enhancements Inc. Unveils New Products
0.60  01-Oct-90  KeyCorp Successful Bidder to Acquire New York State Divisions of Empire of America Federal
0.60  15-Feb-90  Health: U.S. Birth Control Lags
0.45  01-Oct-90  Top Selling Microsoft Windows Applications Now Available in One Convenient Package
```

ウェイトの割り当て

子にウェイトを割り当てるときは、親トピックに対するその子の重要性がウェイトに反映されることに注意してください。一致したドキュメントは、検索に対する重要性によってランク付けされます。選択の結果には、割り当てたウェイトが直接影響します。ウェイトを変更すると、選択の結果も変化します。

例：

boeing-people トピックには、次のように binder、shrontz、woodard という 3 つのウェイト付きの子があります。



これらのサブトピックには、さまざまなウェイトが割り当てられています。binder には 0.80 のウェイトが割り当てられています。これは、この子が 3 つのうち最も重要であると考えられているからです。shrontz サブトピックには「中間」の 0.50 のウェイトが割り当てられています。この子は他の 2 つの子とくらべて比較的重要であると考えられています。woodard サブトピックには 0.30 の低いウェイトが割り当てられています。この子は他の子とくらべて重要性が最も低いと考えられています。

boeing-people トピックを検索に使用する場合、Verity 検索エンジンでは、“Paul Binder” というフレーズがドキュメント内にある場合、そのドキュメントは boeing-people トピックを使用する検索に関連がある可能性が高いと想定されます。“Frank Shrontz” というフレーズを含むドキュメントはこの検索に比較的関連し、“Ron Woodard” というフレーズを含むドキュメントの関連は最も小さくなります。

boeing-people トピックには ACCRUE 演算子が割り当てられているため、結果リストの上部に表示されるドキュメントは、含まれる子の数が最も多いドキュメントになります。したがって、この 3 人すべてについての参照を含むすべてのドキュメントの重要性が最も高くなります。1 つの名前だけを含むドキュメントは、それぞれの子のウェイトを反映した順序で選択されます。したがって、binder という子に最も大きいウェイトがあるため、1 人だけを含むドキュメントの場合は、まず Paul Binder、次に Frank Shrontz、最後に Ron Woodard を参照するドキュメントの順にランク付けされます。

自動ウェイト割り当て

子を作成すると、Verity 検索エンジンは、ACCRUE 演算子を使用するトピックの子に対してデフォルトの 0.50 のウェイトを自動的に割り当てます。1.00 のウェイトは、AND または OR 演算子を使用するトピックの子に自動的に割り当てられます。「ウェイトの変更」(112 ページ) で説明するように、これらのデフォルトのウェイトは手動で増減させることもできます。近接演算子を使用するトピックから形跡トピックを作成する場合は、1.00 というデフォルトのウェイトが割り当てられます。このウェイトは変更できません。

ウェイト割り当てのヒント

最初にウェイトを割り当てるときは、ACCRUE トピックの子に 0.50、その他すべてのトピックの子に 1.00 のウェイトを割り当てることから始めます。

ACCRUE 演算子を使用するトピックの子にウェイトを割り当てるとき、その子に過度に高いウェイトがない場合は、より関連の強い結果を選択できます。たとえば、ACCRUE トピックのすべての子に 1.00 のウェイトを割り当てると、ドキュメント内に子がいくつあるかに関係なくすべてのドキュメントの重要性が等しくなります。Verity 検索エンジンは、すべての子を含むドキュメントに対してだけでなく、1 つしか子を含まないすべてのドキュメントに対しても等しい重要性を割り当てるので、選択結果が表示されたときこれらのドキュメントを区別できません。

最適な選択結果を得るには、0.80 から 0.20 までのウェイトを割り当ててください。

ウェイトの変更

子にウェイトを割り当てたら、親トピックを使用した検索を行ってこれらのウェイトをテストし、適切なドキュメントが選択されるかどうか確認します。ウェイトを変更する必要性が認められた場合は、そのサブトピックまたは形跡トピックに割り当てられた既存のウェイトを編集できます。トピックのアウトライン・ファイルでトピック定義を編集するときは、mktopics を使用してトピック・セットを再構築する必要があることに注意してください。mktopics の使用方法の詳細については、お使いの Verity アプリケーションの管理者ガイドを参照してください。

トピックのスコア計算とドキュメントの重要性

トピックを使用して検索する場合、検索エージェントはそのトピックの形跡トピックから分析を始めます。形跡トピックが存在する場合、スコアは 1.00 になり、検索に関連するとみなされます。形跡トピックが存在しない場合、スコアは 0.00 になり、検索とは無関連とみなされます。形跡トピックがウェイト付きの場合は、形跡トピックのスコアがウェイトによって乗算され、それぞれの積が、親トピックの演算子によって指定された方法で結合されます。逆にこの親トピックが、別の検索対象トピックの子である場合、そのスコアは割り当てられたウェイトによって乗算され、その積は、この親トピックに割り当てられた演算子によって指定された方法で他の子の積に結合されます。このプロセスは、親トピックに達するまで継続されます。

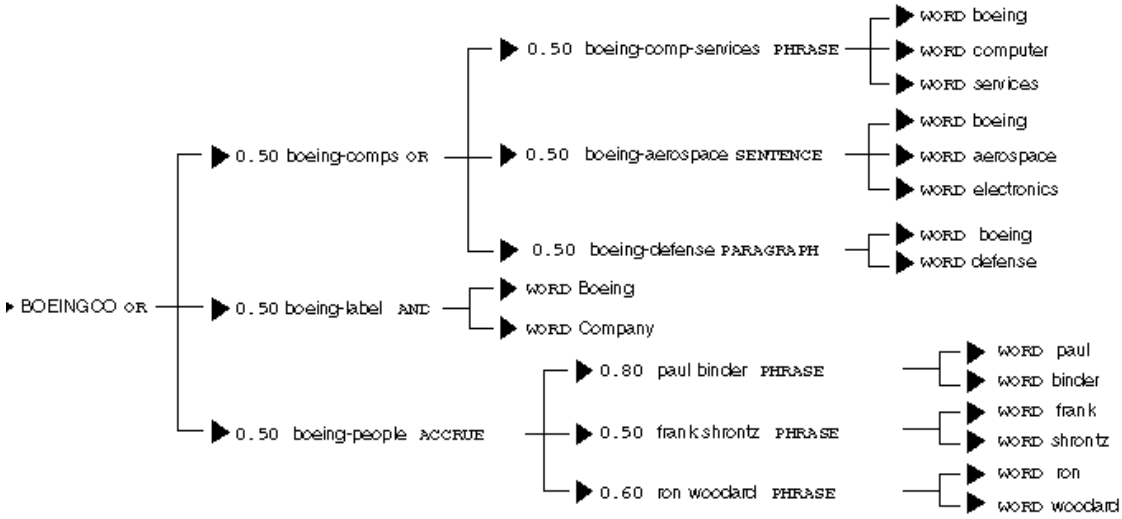
使用する演算子によって、選択されたドキュメントの重要性に親と子のスコアがどのように影響するかが決まります。トピック内のそれぞれの子に重要性スコアが付けられるときは、次のような計算が行われます。

- トピックが ACCRUE 演算子を使用した場合、最も高いランク結果はそれぞれの子のウェイトとスコアの積から取られ、次にドキュメント内に存在する子それぞれのスコアに少し追加されます。
- トピックが AND 演算子を使用した場合、それぞれの子のウェイトとスコアの積が比較され、最も低い積 (最小) がスコアとして取られます。
- 子が OR 演算子を使用した場合、それぞれの子のウェイトとスコアの積が比較され、最も高い積 (最大) がスコアとして取られます。
- 子が近接演算子 (PHRASE、SENTENCE、または PARAGRAPH)、または関係演算子を使用した場合、子のスコアはトピックが存在する場合は 1.00、トピックが存在しない場合は 0.00 になります。
- 形跡トピックのスコアは、このトピックが存在する場合は 1.00、存在しない場合はスコアなしの 0.00 になります。

親トピックの最後の計算が終了すると、一致したドキュメントが Verity アプリケーションで使用可能になり、ユーザにはそのドキュメントが強調表示されます。

次の例では形跡トピックとサブトピックの計算方法を分解し、選択されたドキュメントに重要性を割り当てるプロセスを示しています。

次の図では親トピックの BOEINGCO が検索で使用されます。



各サブトピックの形跡トピックはドキュメントに対して最初にチェックされ、存在するかどうかが判別されます。存在する形跡トピックには 1.00 のスコアが、存在しない形跡トピックには 0.00 のスコアがそれぞれ割り当てられます。

トピック構造の次のレベルにある演算子は、形跡トピックのスコアを結合するのに使用します。このレベルの演算子はすべて近接演算子であるため (したがって、ウェイトは割り当てられていないため)、これらはすべて、0.00 か 1.00 のスコアを生成します。

たとえば、次のような形跡トピックがあるドキュメント内に現れるとします。

- 形跡トピック “Boeing Computer Services” がフレーズ内に現れる。
- 形跡トピック “Boeing Defense” が段落内に現れる。形跡トピック “Boeing Company” がドキュメント内に現れる。
- 形跡トピック “Ron Woodard” がフレーズ内に現れる。

その他の形跡トピックは、部分的にのみ存在するか、または存在しません。表 8-9 は、これらの形跡トピックの有無がトピックのスコアにどのように影響するかを示します。各トピックのスコアは関連するすべての形跡トピックの存在を反映していますが、これは親トピックに割り当てられた演算子に基づいています。

表 *8-9:* 形跡トピックとスコア

トピック名	形跡トピック	形跡トピックが存在する	形跡トピックが存在しない	トピックのスコア
boeing-comp-services	boeing computer services	1		1
		1		
		1		
boeing-aerospace	boeing aerospace electronics	1	1	0
			1	
boeing-defense	boeing defense	1		1
		1		
boeing-label	boeing company	1		1
		1		
paul-binder	paul binder		1	0
			1	
frank-shrontz	frank shrontz		1	0
			1	
ron-woodard	ron woodard	1		1
		1		

トピックのスコアが上記の場合、構造内の次のレベルのトピックにある演算子は次のように計算されます。

- サブトピック boeing-comps は AND 演算子を使用し、スコアは 0.50 になる。
- サブトピック boeing-people は ACCRUE 演算子を使用し、スコアは 0.50 になる。

最後に、トピック BOEINGCO は OR 演算子を使用し、それぞれの子のウェイトとスコアの積を比較し、最も高い積 (最大) をスコアとして取ります。したがって、選択されるドキュメントのスコアは 0.50 になります。

このプロセスが各ドキュメントに繰り返されます。ドキュメントは BOEINGCO トピックのスコアによってソートされ、ランク付けされた順序で表示されます。

トピックの設計

この項では、効率的なトピックを設計する方法について説明します。ここで説明する方法は、トピックのアウトライン・ファイルや Verity クライアントの 1 つを使用してトピックを構成するときに適用することができます。このセクションで説明する情報は、次のとおりです。

- トピック設計の準備
- トピックの設計方式
- 初期トピックの設計

トピック設計の準備

トピックの設計を準備するときには、使用する命名規則を考慮してください。トピックの名前は、検索するドキュメントの内容を識別するのに役立ちます。

最善の検索パフォーマンスを確保するには、トピックの名前に英数字 (A から Z までと 0 から 9 まで) を使用します。また、ASCII 値が 128 以上の各国文字、$ (ドル記号)、% (パーセント記号)、^ (曲折アクセント記号)、+ (プラス記号)、- (ダッシュ)、_ (アンダースコア) も使用できます。その他の非英数字を使用すると、トピックの名前が誤って解釈されて検索結果に影響する場合があります。

情報のニーズの理解

トピック設計の対象となるサブジェクト領域と、自己のサイトでのユーザの検索要件について理解しておいてください。その次に、情報のニーズと、検索するドキュメントのタイプについて理解します。

初期トピックの設計を考案する際には、設計方式 (戦略) を策定し、定義するトピックはその戦略を実現するための戦術であるということに留意してください。

設計方式を策定するときには、次の問題に答えてください。

- 何を検索するために Verity 検索エージェントを使用するか ?
- Verity 検索エージェントによってどのような問題を解決するか ?
- 検索エージェントを使用するのは誰か ?
- どのような種類のソースを使用するか ?
- どのような種類の検索を行うか ?
- 現在はどのように検索を行っているか ?

定義するトピックは、投じられる質問と考えてください。ちょうど、サブジェクト領域に関連する情報を近くの図書館の館員にたずねるのと同様に、作成するトピックは Verity 検索エージェントを作成するときに質問をします。

戦略を考案し、Verity 検索アプリケーションをどのように実装して問題を解決するかを考えるとき、設計するトピックは次のようないくつかの役割を果たすことに注意してください。

- 図書館員
- リサーチ・アシスタント
- 情報のレポジトリ
- 知識ベース

ドキュメントの理解

効率的なトピックを構築するには、情報ソースとして使用するドキュメントのタイプをよく理解しておく必要があります。たとえば、ドキュメントには次のようなタイプの情報が 1 つまたは複数含まれています。

- 手紙
- メモ
- レポート (R)
- アーティクル

検索するドキュメントのタイプの代表的なサンプルを収集します。設計するトピックに適用する必要がある一般的な特性をメモします。たとえばドキュメントに重要な用語、頭字語、または専門用語が含まれている場合は、このテキストを検索するトピックを作成できるようにそれらを強調します。

ドキュメントのサンプルを収集しながら、そのソースを識別します。つまり、内部監査レポートなどの内部ソースであるか、または外部組織からの電子メール・メッセージなどの外部ソースであるかを判別します。この情報により、最上位トピックのサブトピックを定義できます。

スキャン・データの使用

OCR 機能を使用してドキュメントを電子ファイルにスキャンする場合は、インデックス作成の前にドキュメント・ファイルの正確さを確認するかどうかを決定します。スキャンしたファイルを確認する場合は、確認担当者と相談して、用語、頭字語、専門用語が標準化されるようにします。スキャンしたファイルを確認しない場合は、バリエーションが生じる可能性に注意します。OR 演算子を使用するトピックを開発すると、バリエーションを含めることができます。

ドキュメントのサンプルの分類

代表的なドキュメントのサンプル収集と、その内容の最初の分析が終わったら、これらのサンプルをさらに分類できます。この分類プロセスにより、トピック設計に含まれる最上位トピックとその子を定義でき、また割り当てる演算子とウェイトを判断できます。

分類の例は次のとおりです。

- 地理的な場所
- サイト
- プロジェクト
- サブジェクト領域
- 日付

この分類プロセスにより、情報ソースに存在する一般的で意味のある要素を理解できます。たとえば、情報を日付 (月など) で分類する場合は、EQUALS などの関係演算子を使用するトピックを作成するのが適当です。

トピックの設計方式

ドキュメントについて理解したら、トピックの設計方式を選択します。トピックの設計方式には次の 2 つがあります。

- 「トップダウン」方式は、主要なサブジェクトの分類を最初に行い、次に増加する詳細の分類を行う方法です。
- 「ボトムアップ」方式は、詳細な領域を最初に分類し、次により一般的なサブジェクトによって、各詳細領域をグループ分けする方法です。

トップダウン設計

トップダウン方式では、最上位トピックから各サブトピックの個々の形跡トピックへと、上から下に向かってトピックを設計していきます。トップダウンで設計するには、次のように分類法や科学的な分類方式を取り入れてトピックを作成する必要があります。

- 最上位トピック － 一般的な見出しによってサブジェクト領域を識別する
- サブトピック － より限定的な見出しによって、徐々に詳細になっていくトピックだけでなく、サブジェクト領域内での基本的なグループ化も識別する
- 形跡トピック － 重要な用語、頭字語、または専門用語によって、サブジェクトを定義する

トップダウン設計が最もよく機能するのは、明確に定義された必要条件がある場合です。この方法は、検索可能なドキュメントのセットが常に増加または変化している場合にも最適です。この方式は、たとえば情報ソースにまだ形跡がないかもしれないサブジェクトを定義する場合に使用します。いくつかの新規ドキュメントに、トピック設計で識別されていない情報が含まれていることがわかった場合は常に、新しいトピックを追加できることに注意してください。

情報ソース (インデックスを作成したドキュメントのセット) が常に変化している場合は、ドキュメント内の特定のサブジェクト、特に最下位レベルのサブジェクトが検出されない場合があります。トピックによって選択される情報を定期的に分析し、アプリケーションに対して重要なトピックが最新であり、適切な情報が検出されていることを確認してください。

ボトムアップ設計

ボトムアップ方式では、個々の形跡トピックから、定義される最上位トピックに至るまでの順でトピックを設計することが前提となります。この方式では、トピック設計の目的は最下位トピックに類似した情報を含むドキュメントを選択することです。

ボトムアップ設計を行う場合は、検索するワードまたはフレーズの代表的なサンプルを含むドキュメントから始めます。次に、連続した上位の分類によってこれらのワードをグループ化します。

ボトムアップ設計が最もよく機能するのは、一部のドキュメントが、類似の情報を含むその他の多くのドキュメントの代表となるような場合です。この方法は、情報ソースが多くの変更や追加に影響されないときにも有効です。

特定のドキュメントの内容に基づいたトピック設計では、他のドキュメント内にある関連のサブジェクト領域が検出されない場合があります。たとえば、サンプル・ドキュメントで使用されている名前が他のドキュメントで変更されると、その新しい名前は検索ではヒットしない場合があります。

さらにボトムアップ方式では、トピックの開発に使用する特定のドキュメント・セットに合わせたトピック設計になります。この特定のドキュメントは、情報ソース内にあるすべてのドキュメントを代表するものではない場合もあります。検索の有効性は定期的に確認してください。

初期トピックの設計

初期トピックの設計方式として、トップダウン方式とボトムアップ方式のいずれを使用するかを決定したら、定義するトピックのレベルを確認するためにトピックのアウトラインを作成します。

トピックのアウトラインの作成

トピックのアウトラインを作成すると、トピック内の種々のレベルで情報をどのように分類するかを決めるのに役立ちます。トピックのアウトラインはトップダウン方式とボトムアップ方式のいずれの設計方法でも使用できますが、特に有効なのはトップダウン方式です。作成する 1 つ 1 つのトピックをまず、アウトラインとして開発することをおすすめします。こうすると、トピックとサブトピックの関係を理解して、最も使いやすく編成することができます。

トピックのアウトラインによって、サイトで Verity 検索エージェントを使用する人がどのように情報を検索するかを理解することができます。トピックのアウトラインを使用すると、トピックとサブトピックによって指定される情報を微調整し、ドキュメントを正確に選択できます。トピックのアウトラインを開発するときは、次のことを行ってください。

- ユーザが検索を実行するときに使用する特定の情報分野を確認する。
- 親トピックの下に子としてグループ化できる関連サブトピックをすべて確認する。
- トピック設計が扱う詳細の最初のレベルを考える。

トピックのアウトラインがカバーする範囲は、最初は比較的小さくしてください。トピックのアウトラインは小さく単純な方が定義しやすく、また後からいつでも情報を追加できます。トピックのアウトラインを開発しながら、トピック設計に含まれるレベルの数を決定します。

トップダウンのアウトラインの例

トップダウンのアウトラインの開発には、次の 3 つの手順があります。

- 情報階層の設定
- 個々の検索カテゴリの設定
- 構築するトピックの設定

この 3 つの手順を行いながら、サイトで Verity 検索エージェントを使用する人々と出会い、検索ニーズに最適なアウトラインを開発します。以下ではこの手順について説明します。

情報階層の設定

サイトで出会う人々と話し、ユーザが求める情報がどのようなタイプのドキュメントに含まれているかを知ります。

たとえば、医療産業に携わる人のために、最新の医薬品試験に関する情報を検索するトピックを設計すると仮定します。サイトで Verity 検索エージェントを使用する人との話し合いから、次のようなドキュメントが、最新の医薬品試験情報の最重要ソースであることがわかります。

- リサーチ・レポート
- 製品文献

これらのドキュメントが、Verity 検索エージェントで検索する情報ソースを構成します。

個々の検索カテゴリの設定

サイトの情報ソースを構成するドキュメントを確認します。ドキュメントを分類する方法を考えます。

この例では、医療リサーチ・レポートと製品文献の確認によって、これらのドキュメントに含まれる情報が明らかになり、それをいくつかのカテゴリに分割します。次のカテゴリを使用して、トピック設計の最上位トピックを定義することを決定します。

- 研究レポート
- 臨床上の試験、データ、またはリサーチ
- 製品文献

構築するトピックの設定

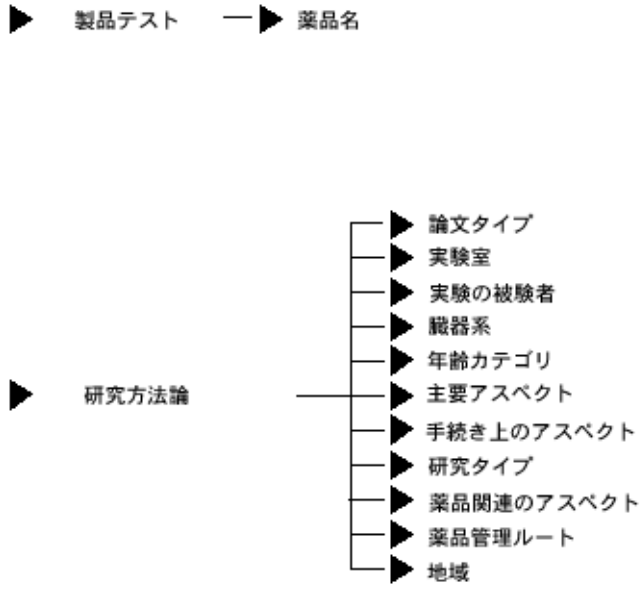
定義するカテゴリについて、サイトで Verity 検索エージェントを作成する人と話し合い、選択されたドキュメントに含まれている必要がある最も重要な概念、および各カテゴリに開発する必要がある最上位トピックを決定します。

たとえば、“clinical trials” (臨床試験) カテゴリには次の最上位トピックが入るようにします。

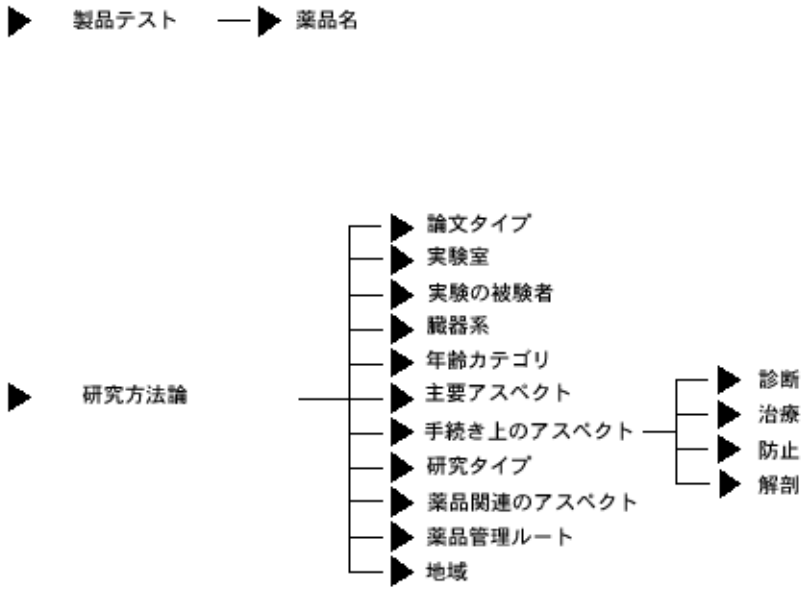
製品テスト

研究方法論

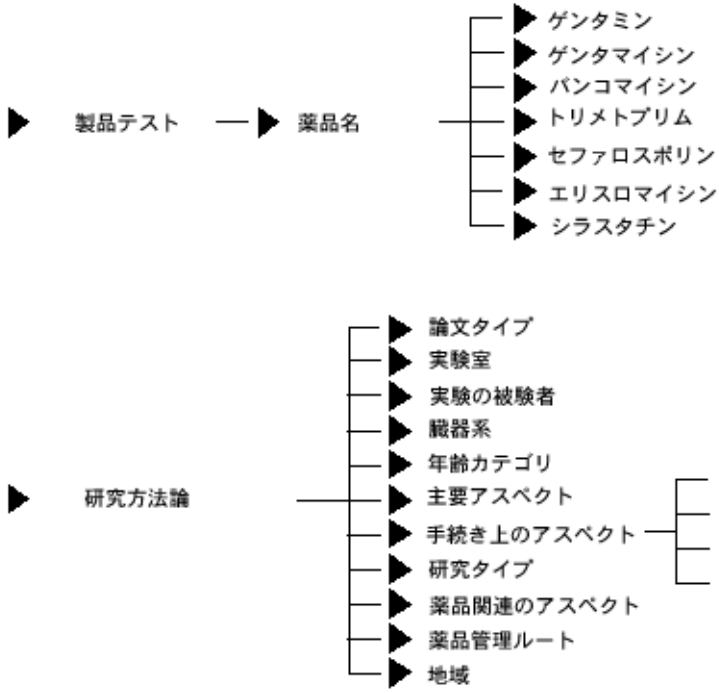
最上位トピック内では、たとえば、次のようなサブトピックがサブジェクト領域のエキスパートによって確認されます。



これらのトピックの分類が終わったら、サイトで Verity 検索エージェントを使用する人に相談してサブトピックを決定します。次は、トピックの手続きの面を示す子として分類されたサブトピックの例です。



トピックのアウトラインを定義しながら、サイトで Verity 検索エージェントを使用する人々と相談して、トピックが意味のあるドキュメントを選択するようにします。次の例では、drug-names というトピックによって、ユーザが医薬品の名前に基づいて臨床試験データを検索できます。



ボトムアップのアウトラインの例

ボトムアップのアウトラインの開発には、次の 3 つの手順があります。

- トピック設計の最下位を構成するサブトピックの識別
- 関連するサブトピックの上位トピックへの分類
- 最上位トピックの分類の設定

この 3 つの手順を行いながら、サイトで Verity 検索エージェントを作成する人々と出会い、検索ニーズに最適なアウトラインを開発します。以下ではこの手順について説明します。

最下位トピックの確認

検索する他のドキュメントを代表するような情報を含んでいる、モデルとして使用できるドキュメントを探します。

たとえば、コンピュータ産業についての情報を検索するトピック設計を行っているとします。最初に、Apple Computer と関連製品についてのドキュメントを検索するトピックを構築します。

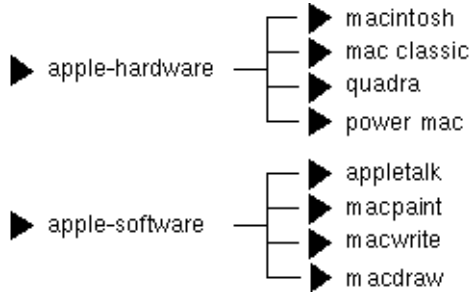
次の例を、検索する他のドキュメントを代表するような情報を含むモデル・ドキュメントとして使用します。

```
A system developed specifically for networked Apple Computer, Inc.
Macintosh computers has been announced by Human Designs, Inc.

Dubbed Chorus, the floor-standing unit reportedly can contain up to
16 floating-point processors and connects to networked Macintoshes to
create a multiuser desktop environment.

The product offers performance of eight million to 32 million
floating-point operations per second  and was designed to accommodate
software development, according to the vendor.  Options include an
Ethernet I/O upgrade and a software simulator.

A Chorus 1 single floating-point processor entry-level system costs
$9,700. A Chorus 4 configuration with four floating point processors
is available at $25,000, which includes a dedicated I/O processor
with an Apple Appletalk port and system software.  Both systems are
upgradable.

Human Designs, 322 W. 71st St., New York, N.Y. 10023. 212-580-0257.
```

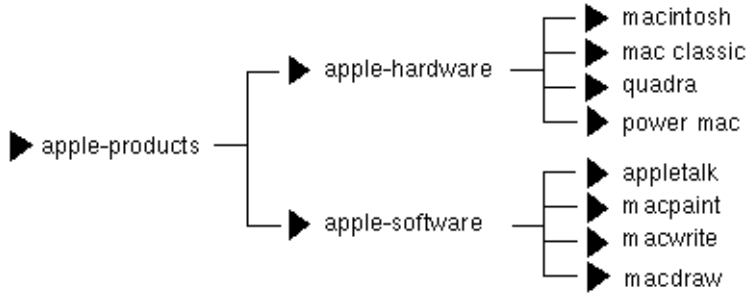
このドキュメントによって、“Appletalk” と “Macintosh” について言及している他のドキュメントを見つけることが決まるので、apple-software と apple-hardware という 2 つの親トピックの名前を定義します。

形跡トピックを追加して、関連する情報を含むドキュメントを選択できるようにします。関連する情報とはたとえば “Macintosh”、

“Mac Classic”、“Quadra”、“Power Mac” などです。さらに、形跡トピック “AppleTalk”、“MacPaint”、“MacWrite”、“MacDraw” を関連ソフトウェア製品として含めることにします。これらの形跡トピックを、apple-hardware トピックと apple-software トピックに次のように割り当てます。



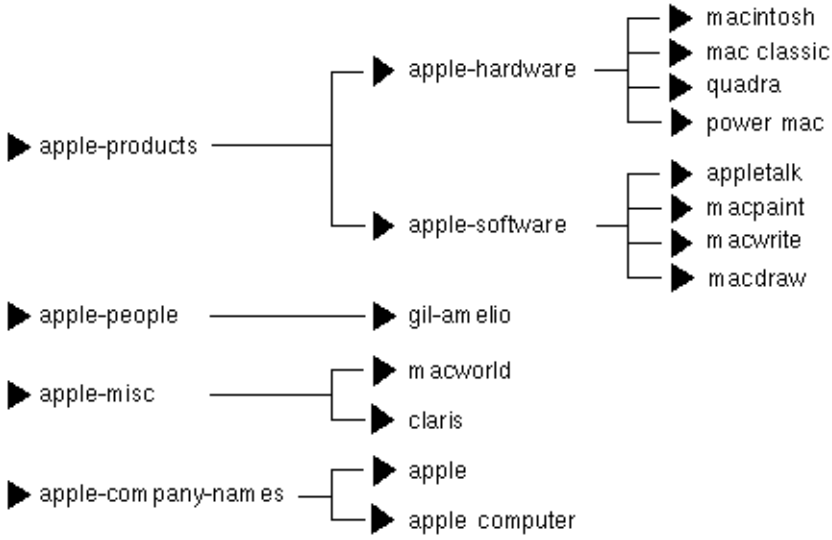
最後に、これらのトピックを次のように apple-products トピックとして統合します。



関連するサブトピックの分類

サイトで Verity 検索エージェントを使用する人とサブトピックについて話し合い、カテゴリ内で論理的にグループ化できるその他のサブトピックが存在するかどうかを判断します。

この例では、Verity 検索エージェントを使用する人々には、Apple Computer の社員についての情報検索に関心を持っている人もいれば、Apple Computer について言及しているドキュメントの検索に関心を持っている人もいます。次の例では、トピックの論理グループが Apple Computer のいくつかの面を表しています。



最上位トピックの設定

関連情報の検索のために他の最上位トピックが必要かどうかを判断します。

次の例では、新しいトピックの dec を、別のコンピュータ メーカーの Digital Equipment Corporation のために開発しています。次に示すように、このトピックは最上位トピックに割り当てられ、apple トピックに定義したのと同様のサブトピックが含まれています。

付 録　A

システム・プロシージャ

この付録では、Sybase が提供するシステム・プロシージャについて説明します。これらのシステム・プロシージャは、システム・テーブルを更新したりシステム・テーブルからレポートを取得したりするために使用されます。表 A-1 は、拡張型全文検索エンジンに備わっているシステム・プロシージャの一覧です。

表 *A-1:* システム・プロシージャ

プロシージャ	説明
sp_check_text_index	拡張型全文検索インデックスとソース・テーブル内の一貫性の問題をレポートまたは修復する。
sp_clean_text_events	処理済みのエントリを text_events テーブルから削除する。
sp_clean_text_indexes	テーブルと対応していないテキスト・インデックスを削除する。
sp_create_text_index	外部テキスト・インデックスを作成する。
sp_drop_text_index	テキスト・インデックスを削除する。
sp_help_text_index	テキスト・インデックスを表示する。
sp_optimize_text_index	Verity の最適化ルーチンを実行する。
sp_redo_text_events	text_events テーブル内のエントリのステータスを変更し、修正されたテーブルのインデックスを強制的に付け直す。
sp_refresh_text_index	対応する送信元テーブルへの更新を反映しながら、text_events テーブルにエントリを追加する。
sp_show_text_online	現在オンラインにあるデータベースまたはインデックスの情報を表示する。
sp_text_cluster	クラスタ・オプションを表示または修正する。
sp_text_configure	拡張型全文検索エンジンの設定パラメータを表示または修正する。
sp_text_dump_database	データベース内にテキスト・インデックスのバックアップ・コピーを作成する。また、オプションとして text_db データベースと現在のデータベースをダンプする。
sp_text_kill	特定のテキスト・インデックスへの接続をすべて終了する。
sp_text_load_index	バックアップからテキスト・インデックスをリストアする。
sp_text_notify	text_events テーブルが修正されたことを拡張型全文検索エンジンに通知する。
sp_text_online	Adaptive Server からデータベースを利用できるようにする。

sp_check_text_index

説明

拡張型全文検索インデックスとソース・テーブル内の一貫性の問題をレポートまたは修復します。

構文

sp_check_text_index *server*, "*index_name*", "id_column", "fixit"

パラメータ

server
テキスト・サーバの名前。

index_name
テキスト・サーバの名前。

id_column
送信元 identity カラムまたはプライマリ・キー・カラムの名前。

fixit
FALSE の場合、問題をレポートします。TRUE の場合、問題をレポートしないで修復します。

例

text.i_text という名前のカラムの問題を textsvr という名前のサーバでリストします。

```
sp_check_text_index "textsvr", "text.i_text", "id", "false"
```

使用法

- sp_dboption “select into”, true を発行してから sp_check_text_index を使用してください。
- このプロシージャは、次の処理を実行します。
  - エントリが送信元テーブルにあっても、インデックス内に一致するものがない場合、それらのエントリに対する sp_refresh_text_index を生成する。
  - エントリがインデックス・テーブルにあっても、送信元テーブルにない場合、それらのエントリに対する sp_refresh_text_index delete を生成する。
  - エントリがインデックス内に重複して存在する場合、余分な各エントリに対する sp_refresh_text_index delete を生成する。
- インデックスの重複を確認するには、インデックス・テーブルの ID の値をすべて選択して、テンポラリ・テーブルに入れます。コレクションに 64K 個より多くの ID 値がある場合、batch_blocksize 設定パラメータをデフォルトの 0 から 65536 に変更して、返される Verity 情報のブロック読み込みを有効にする必要があります。ブロック読み込みを有効にしない場合、拡張型全文検索はすべての ID 値を一度に読み込もうとするため、“-27” の Verity エラーで失敗します。

メッセージ

なし

パーミッション

すべてのユーザが、sp_check_text_index を実行できます。

sp_clean_text_events

説明　処理済みのエントリを text_events テーブルから削除します。

構文　sp_clean_text_events [*up_to_date*]

パラメータ　*up_to_date*
処理済みのすべてのエントリを削除する最後の日時。

例　1998 年 1 月 15 日の午後 5 時以前に入力されたデータを削除します。

```
sp_clean_text_events "01/15/98:17:00"
```

使用法

- *up_to_date* パラメータを指定した場合、*up_to_date* 以前の日時を持ちステータスが処理済みに設定されているすべてのエントリが削除されます。
- *up_to_date* を省略した場合、ステータスが処理済みに設定されているすべてのエントリが削除されます。
- テキスト・インデックスと対応するコレクションを必ずバックアップした後で、エントリを text_events テーブルから削除します。
- sp_text_dump_database は自動的に実行されます。

メッセージ　なし

パーミッション　すべてのユーザが、sp_clean_text_events を実行できます。

参照　sp_text_dump_database

sp_clean_text_indexes

説明 vesaux テーブルからテーブルと対応していないインデックスを削除します。

構文 sp_clean_text_indexes

パラメータ なし

例

```
sp_clean_text_indexes
```

使用法

- このプロシージャは、送信元テーブルおよび対応するインデックス・テーブルが存在することを確認しながら、vesaux と vesauxcol テーブルからエントリを読み込みます。どちらかが存在しない場合、そのインデックスは削除されます。

メッセージ

- Fetch resulted in an error.
- Unable to drop object definition for *index_name*!

パーミッション すべてのユーザが、sp_clean_text_indexes を実行できます。

sp_create_text_index

説明

テキスト・インデックスを作成します。

構文

sp_create_text_index *server_name*, *index_table_name*,
table_name, "batch", *column_name*
[, *column_name* ...]

パラメータ

server_name
拡張型全文検索エンジンの名前。

index_table_name
インデックス・テーブルの名前。*index_table_name* は [dbname.[owner.]]table の形式をとります。各パラメータの意味は、次のとおりです。

- dbname はインデックス・テーブルが入っているデータベースの名前です。
- owner はインデックス・テーブルの所有者の名前です。
- table はインデックス・テーブルの名前です。

table_name
インデックスを作成するテキストが入っている送信元テーブルの名前。*table_name* は [dbname.[owner.]]table の形式をとります。

batch
“batch” 演算子 (引用符が必要) は、VDK にバッチが送信されるたびにすべてのセッションを再割り付けするよう、拡張型全文検索に指示します。

column_name
テキスト・インデックスを作成するカラムの名前。

例

blurbs テーブルの copy カラムに対するテキスト・インデックスと i_blurbs という名前のインデックス・テーブルを作成します。

```
sp_create_text_index "blue", "i_blurbs", "blurbs", " ", "copy"
```

使用法

- 単一のテキスト・インデックス内に最大 16 カラムのインデックスを作成できます。
- インデックスを作成できるカラムのデータ型は、char、varchar、nchar、nvarchar、text、image、date、time、datetime、smalldatetime、int、smallint、tinyint です。
- *option_string* では大文字と小文字が区別されません。
- *option_string* で「オプションなし」を指定するには null 文字列 (" ") を使用します。

- すぐに削除するテキスト・インデックスを作成するには *option_string* に “empty” という値を代入します。これにより、Verity コレクションのディレクトリとスタイル・ファイルが作成されますが、ディレクトリ内にコレクションは作成されません。たとえば、クラスタ用に個々のテーブルを設定する場合、テキスト・インデックスを作成し、すぐに削除します。*style.prm* ファイルを編集した後で、もう一度テキスト・インデックスを作成します。詳細については、「個々の style.prm ファイルの編集」(27 ページ) を参照してください。
- sp_create_text_index は vesaux テーブルにエントリを書き込みます。次に、テキスト・インデックスを作成するよう拡張型全文検索エンジンに指示を出します。
- sp_create_text_index の実行には、同期がとられます。このシステム・プロシージャを実行している Adaptive Server プロセスは、インデックスの作成が終わるまでブロックされます。大量のデータに対するインデックスを作成する場合、完了までに数時間かかることもあります。
- 複数のカラムに対するテキスト・インデックスを作成する場合、テキスト・インデックス内の各カラムはそれぞれのドキュメント・ゾーンに置かれます。ゾーンの名前は、カラムの名前と同じです。ゾーンを使用して検索の対象を特定のカラムに制限できます。詳細については、「in」(51 ページ) を参照してください。
- インデックスを作成した後で、その名前を変更しないでください。

メッセージ

- Cannot run sp_create_text_index from within a transaction.
- ‘*column_name*’ cannot be NULL.
- Column ‘*column_name*’ does not exist in table ‘*table_name.*’
- Index table mapping failed - text index creation aborted.
- Invalid text index name - ‘*index_name*’ already exists.
- ‘*parameter*’ is not in the current database.
- Server name ‘*server_name*’ does not exist in sysservers.
- ‘*table_name*’ does not exist.
- ‘*table_name*’ is not a valid object name.
- Table ‘*table_name*’ does not have an identity column - text index creation aborted.
- Text index creation failed.
- User ‘*user_name*’ is not a valid user in the database.

パーミッション

すべてのユーザが、sp_create_text_index を実行できます。

sp_drop_text_index

説明

インデックス・テーブルとテキスト・インデックスを削除します。

構文

sp_drop_text_index "*table_name.index_table_name*"
[,"*table_name.index_table_name*"...]

パラメータ

table_name

削除するテキスト・インデックスと対応するテーブルの名前。table_name は [dbname.[owner.]]table の形式をとります。各パラメータの意味は、次のとおりです。

- dbname はテーブルが入っているデータベースの名前です。
- owner はテーブルの所有者の名前です。
- table はテーブルの名前です。

index_table_name

削除するインデックス・テーブルとテキスト・インデックスの名前。index_table_name は [dbname.[owner.]]index の形式をとります。

例

blurbs テーブルと対応する、インデックス・テーブルとテキスト・インデックスを削除します。

```
sp_drop_text_index "blurbs.i_blurbs"
```

使用法

- まず、sp_drop_text_index　は拡張型全文検索エンジンにリモート・プロシージャ・コール (RPC) を発行して、Verity コレクションを削除します。次に、対応するエントリを vesaux テーブルと vesauxcol テーブルから削除し、インデックス・テーブルを削除し、インデックス・テーブルのオブジェクト定義を削除します。
- 単一の sp_drop_text_index 要求で、最大 255 個のインデックスを指定できます。
- *database* と *owner* を指定しない場合、現在の所有者と現在のデータベースが使用されます。

メッセージ

- Cannot run sp_drop_text_index from within a transaction.
- Index '*index_name*' is not a text index.
- '*parameter_name*' is not a valid name.
- Server name '*server_name*' does not exist in sysservers.
- Unable to drop index table '*table_name*'. This table must be dropped manually.
- User '*user_name*' is not a valid user in the '*database_name*' database.
- vs_drop_index failed with code '*code_name*'.

パーミッション

すべてのユーザが、sp_drop_text_index を実行できます。

sp_help_text_index

説明

現在のデータベースのテキスト・インデックスのリストを表示します。

構文

sp_help_text_index [*index_table_name*]

パラメータ

index_table_name
表示するテキスト・インデックスの名前。

例

例 1 すべてのインデックスを表示します。

```
sp_help_text_index
```

例 2 テキスト・インデックス i_blurbs についての情報を表示します。

```
sp_help_text_index "i_blurbs"
```

使用法

- sp_help_text_index は Enhanced Full-Text Search Specialty Data Store でのみ利用可能です。
- *index_table_name* パラメータを指定した場合、テキスト・インデックスについての情報が表示されます。この情報には、テキスト・インデックスの名前、インデックスの Verity コレクションの名前、送信元テーブルの名前、IDENTITY カラムまたはプライマリ・キー・カラムの名前、インデックスを作成した拡張型全文検索エンジンの名前などが含まれます。
- *index_table_name* を省略した場合、現在のデータベースに入っているすべてのテキスト・インデックスのリストが表示されます。

メッセージ

- No text indexes found in database '*database_name*.'
- Text index '*index_name*' does not exist in database '*database_name.*'
- Object must be in the current database

パーミッション

すべてのユーザが、sp_help_text_index を実行できます。

sp_optimize_text_index

説明　テキスト・インデックスに対して最適化を実行します。

構文　sp_optimize_text_index *index_table_name*

パラメータ　*index_table_name*

最適化するテキスト・インデックスの名前。*index_table_name* は [dbname.[owner.]]table の形式をとります。各パラメータの意味は、次のとおりです。

- dbname はインデックス・テーブルが入っているデータベースの名前です。owner またはプレースホルダがある場合、それも指定する必要があります。
- owner はインデックス・テーブルの所有者の名前です。
- table はインデックス・テーブルの名前です。

例　テキスト・インデックス i_blurbs を最適化して、クエリのパフォーマンスを向上させます。

```
sp_optimize_text_index "i_blurbs"
```

使用法

- sp_optimize_text_index は、Enhanced Full-Text Search Specialty Data Store でのみ利用可能です。
- このシステム・プロシージャによって、拡張型全文検索エンジンは Verity の最適化ルーチンを通して、指定されたテキスト・インデックスを実行するようになります。
- sp_optimize_text_index は、Verity の最適化が無効にされた状態 (トレース・フラグ 11 がオンにされた状態) で更新されたテキスト・インデックスを最適化する場合に役立ちます。
- MaxClean 最適化を有効にするには、トレース・フラグ 30 をオンにします。このトレース・フラグは、追加の処理時間を費やし、通常の使用の妨げになる場合があるので、管理時だけに使用します。MaxClean は古くなったコレクション・ファイルを削除する Verity の最適化機能です。

メッセージ

- '*index_table_name*' is not in the current database.
- '*index_table_name*' does not exist.
- Index '*index_table_name*' is not a text index.
- This procedure is not supported against remote server '*server_name*.'

パーミッション　すべてのユーザが、sp_optimize_text_index を実行できます。

参照　「既存のインデックスの更新」(75 ページ)

sp_redo_text_events

説明

text_events テーブル内のエントリのステータスを変更し、修正されたカラムのインデックスを強制的に付け直します。

構文

sp_redo_text_events [*from_date* [,*to_date*]]

パラメータ

from_date
修正するエントリの日付範囲の開始日時。

to_date
修正するエントリの日付範囲の終了日時。

例

1998 年 1 月 5 日午後 5 時から 1998 年 2 月 12 日午前 8 時 30 分までに修正されたカラムのインデックスをもう一度作成します。

```
sp_redo_text_events "01/05/98:17:00", "02/12/98:08:30"
```

使用法

- 現在「処理済み」のステータスを持つ text_events テーブル内のエントリをすべて「未処理」に再設定します。拡張型全文検索エンジンにもう一度インデックスを作成する必要があることを通知します。
- バックアップから Verity コレクションをリカバリした後、テキスト・インデックスを同期させるために役立ちます。このプロシージャは sp_text_load_index の実行中に自動的に実行されます。
- *to_date* を省略した場合、*from_date* と現在時刻の間のエントリで「処理済み」のステータスを持つものはすべて「未処理」に再設定されます。
- *from_date* と *to_date* の両方を省略した場合、text_events テーブル内の「処理済み」のステータスを持つエントリがすべて「未処理」に再設定されます。

メッセージ

- *to_date* cannot be specified without *from_date*.
- You have not specified the full range.

パーミッション

すべてのユーザが、sp_redo_text_events を実行できます。

sp_refresh_text_index

説明

テキスト・インデックスの送信元テーブルのデータが変更されたときに、text_events テーブル内の修正を記録します。

構文

sp_refresh_text_index *table_name*, *column_name*, *rowid*, *mod_type*

パラメータ

table_name
更新する送信元テーブルの名前。table_name は [dbname.[owner.]]table の形式をとります。各パラメータの意味は、次のとおりです。

- dbname はテーブルが入っているデータベースの名前です。
- owner はテーブルの所有者の名前です。
- table はテーブルの名前です。

column_name
更新するカラムの名前。

rowid
変更されたローの IDENTITY カラムまたはプライマリ・キー・カラムの値。

mod_type
変更のタイプを指定します。insert、update、または delete でなければなりません。

例

blurbs テーブルの copy カラムを更新したことを、text_events テーブル内に記録します。更新したローの id は 2.000000 になります。

```
sp_refresh_text_index "blurbs", "copy", 2.000000, "update"
```

使用法

- テキスト・インデックスの一貫性は、ユーザが管理します。インデックスを作成した送信元データを更新した場合は sp_refresh_text_index を必ず実行して、text_events テーブルに変更を反映させてください。これにより送信元データとコレクションの同期が保たれます。sp_text_notify を実行するまでコレクションは更新されません。
- text でないカラムと image でないカラムに対して sp_refresh_text_index を発行するトリガを作成できます。詳細については、「テキスト・インデックスに対する変更の伝達」(21 ページ) を参照してください。

メッセージ

- Column '*column_name*' does not exist in table '*table_name.*'
- Invalid mod_type specified ('*mod_type'*). Correct values:INSERT, UPDATE, DELETE.
- Owner '*owner_name*' does not exist.
- Table '*table_name*' does not exist.
- '*table_name*' is not a valid name.
- Text event table not found.

パーミッション

すべてのユーザが、sp_refresh_text_index を実行できます。

参照

sp_text_notify

sp_show_text_online

説明

現在オンラインにあるデータベースまたはテキスト・インデックスについての情報を表示します。

構文

sp_show_text_online *server_name* [,{INDEXES | DATABASES}]

パラメータ

server_name

要求を送信する拡張型全文検索エンジンの名前。

INDEXES | DATABASES

オンライン・インデックスとオンライン・データベースのどちらについてのデータを要求するかを指定します。デフォルトは INDEXES です。

例

例 1 KRAZYKAT 拡張型全文検索エンジンで現在オンラインにあるすべてのインデックスを表示します。

```
exec sp_show_text_online KRAZYKAT
```

例 2 KRAZYKAT 拡張型全文検索エンジンで現在オンラインにあるすべてのデータベースを表示します。

```
exec sp_show_text_online KRAZYKAT, DATABASES
```

使用法

- sp_show_text_online はリモート・プロシージャ・コールを拡張型全文検索エンジンに発行し、現在オンラインにあるインデックスまたはデータベースについての情報を取得します。
- この結果にデータベースがリストされない場合、sp_text_online を使用して希望するデータベースをオンラインにします。

メッセージ

- sp_show_text_online failed for server *server_name*.
- The parameter value '*value*' is invalid.
- The RPC sent to the server returned a failure return code.
- The second parameter must be INDEXES or DATABASE.

パーミッション

すべてのユーザが、sp_show_text_online を実行できます。

参照

sp_text_online

sp_text_cluster

説明

アクティブ・スレッドに対するクラスタリング・パラメータを表示または変更します。

構文

sp_text_cluster *server_name, cluster_parameter* [, *cluster_value*]

パラメータ

server_name
拡張型全文検索エンジンの名前。

cluster_parameter
クラスタリング・パラメータの名前。表 A-2 に値を示します。

cluster_value
アクティブ・スレッドに対するクラスタリング・パラメータに代入する値。表 A-2 に値を示します。

表 *A-2:* クラスタリング設定パラメータ

***cluster_parameter* の値**	***cluster_value* の値**
cluster_style	使用するクラスタリングのタイプを指定する。有効な値： • fixed − 一定数のクラスタを生成する。クラスタの数は cluster_max パラメータで設定する。 • coarse − 少数の粗いクラスタに基づいて、生成するクラスタの数を自動的に決定する。 • medium − 中程度のサイズのクラスタに基づいて、生成するクラスタの数を自動的に決定する。 • fine − 小さくて細かいクラスタに基づいて、生成するクラスタの数を自動的に決定する。
cluster_max	cluster_style が fixed に設定されているときに生成するクラスタの最大数を指定する。0 の値は、検索エンジンが生成するクラスタ数を決定することを意味する。
cluster_effort	検索エンジンが適正なクラスタリングに費やす処理量 (時間) を指定する。有効な値： • effort_default − 検索エンジンはデフォルトの時間を費やす。Verity の用語 “default” も、二重引用符 (“ ”) で囲めば使用できる。 • high − 検索エンジンは最も長い時間を費やす。 • medium − 検索エンジンは比較的短い時間しか費やさない。 • low − 検索エンジンは最短時間しか費やさない。
cluster_order	クラスタ内のローを返す順序を指定する。有効な値： • "0" − クラスタの中心に対する類似度に従ってローを返す。これは最初に返されるローがクラスタの最も代表的なローであることを意味する。 • "1" − クラスタに送信されたときと同じ相対順序でローを返す。たとえば、クエリで取り出された 1 番目、3 番目、7 番目のローがクラスタ 1 に格納されている場合、これらのローはクラスタ内のこの相対順序と同じ順序で返される。

例

例 1 アクティブ・スレッドに対する *cluster_order* パラメータを 1 に変更します。

```
sp_text_cluster KRAZYKAT, cluster_order, "1"
```

例 2 *cluster_style* パラメータの現在の値を表示します。

```
sp_text_cluster KRAZYKAT, cluster_style
```

使用法

- Verity のクラスタリングのアルゴリズムは、クラスタリング・パラメータの値に基づいて類似したローのグループ化を試みます。
- *cluster_parameter* パラメータを指定し、*cluster_value* パラメータを省略した場合、sp_text_cluster は指定されたクラスタリング・パラメータの値を表示します。
- sp_text_cluster はクラスタリング設定パラメータの値は修正しません。*cluster_value* は、現在実行されているスレッドに対してだけ有効です。デフォルトの値を修正するには、sp_text_configure を使用します。
- クラスタード結果セットを要求する方法の詳細については、「疑似カラムを使用した、クラスタード結果セットの要求」(46 ページ) を参照してください。

メッセージ

- This procedure is not supported against remote server '*server_name.*'
- The parameter value '*value*' is invalid.
- sp_text_cluster failed (status = *status*).

パーミッション

すべてのユーザが、sp_text_cluster を実行できます。

参照

sp_text_configure

sp_text_configure

説明　拡張型全文検索エンジンの設定パラメータを表示または修正します。

構文　sp_text_configure *server_name* [, *config_name* [, *config_value*]]

パラメータ

server_name
　拡張型全文検索エンジンの名前。

config_name
　表示または修正する設定パラメータの名前。

config_value
　設定パラメータに代入する値。

例

例 1 バックアップ先のディレクトリを */data/backup* に変更します。

```
sp_text_configure KRAZYCAT, backdir, "/data/backup"
```

例 2 バックアップ先のディレクトリを表示します。

```
sp_text_configure KRAZYCAT, backdir
```

使用法

- sp_text_configure を実行して動的なパラメータを修正すると、以下の処理が実行されます。
  - 設定値と実行値が更新されます。
  - 設定ファイルが更新されます。
  - 変更はすぐに有効になる。
- sp_text_configure を実行して静的なパラメータを修正すると、以下の処理が実行されます。
  - 設定値が更新されます。
  - 設定ファイルが更新されます。
  - 拡張型全文検索エンジンが再起動されたときに変更が反映されます。
- パラメータを指定しないで発行すると、sp_text_configure は拡張型全文検索エンジンのすべての設定パラメータとその現在値のレポートを表示します。
- *config_name* パラメータを指定し、*config_value* パラメータを省略した場合、sp_text_configure は指定された設定パラメータのレポートを表示します。
- 個々の設定パラメータの詳細については、「設定パラメータの修正」(62 ページ) を参照してください。

メッセージ

- Configuration value cannot be specified without a configuration option.
- This procedure is not supported against remote server '*server_name*.'
- sp_text_configure failed - possible invalid configuration option '*config_name*.'

パーミッション　すべてのユーザが、sp_text_configure を実行できます。

sp_text_dump_database

説明

テキスト・インデックスのバックアップ・コピーを作成します。

構文

sp_text_dump_database *backupdbs* [, *current_to*] [,
current_with] [, *current_stripe01* [, ... [,
current_stripe31]]] [, *textdb_to*] [, *textdb_with*] [,
textdb_stripe01 [, ... [, *textdb_stripe31*]]]

パラメータ

backupdbs

テキスト・インデックスをバックアップする前に、現在のデータベースと text_db データベースをバックアップするかどうかを指定します。有効な値:

- CURRENT_DB_AND_INDEXES − 現在のデータベースをバックアップしてから、テキスト・インデックスをバックアップする。
- CURRENT_DB_AND_CURRENT_INDEXES − 現在のデータベースをバックアップしてからテキスト・インデックスをバックアップし、現在のデータベースと対応するインデックスだけをダンプする。
- TEXT_DB_AND_INDEXES − text_db データベースをバックアップしてから、テキスト・インデックスをバックアップする。
- INDEXES_AND_DATABASES − 現在のデータベースと text_db データベースをバックアップしてから、テキスト・インデックスをバックアップする。
- ONLY_INDEXES − テキスト・インデックスだけをバックアップする。

current_to

現在のデータベースをダンプする dump database コマンドの to 句。*backupdbs* パラメータで CURRENT_DB_AND_INDEXES または INDEXES_AND_DATABASES を指定する場合のみ、これを使用します。

current_with

現在のデータベースをダンプする dump database コマンドの with 句。*backupdbs* パラメータで CURRENT_DB_AND_INDEXES または INDEXES_AND_DATABASES を指定する場合のみ、これを使用します。

current_stripe

現在のデータベースをダンプする dump database コマンドの stripe 句。*backupdbs* パラメータで CURRENT_DB_AND_INDEXES または INDEXES_AND_DATABASES を指定する場合のみ、これを使用します。

textdb_to

text_db データベースをダンプする dump database コマンドの to 句。*backupdbs* パラメータで INDEXES_AND_DATABASES を指定する場合のみこれを使用します。*backupdbs* パラメータで TEXT_DB_AND_INDEXES または INDEXES_AND_DATABASES を指定する場合のみ、これを使用します。

textdb_with
text_db データベースをダンプする dump database コマンドの with 句。*backupdbs* パラメータで TEXT_DB_AND_INDEXES または INDEXES_AND_DATABASES を指定する場合のみ、これを使用します。

textdb_stripe
text_db データベースをダンプする dump database コマンドの stripe 句。*backupdbs* パラメータで TEXT_DB_AND_INDEXES または INDEXES_AND_DATABASES を指定する場合のみ、これを使用します。

例

例 1 テキスト・インデックスだけがバックアップされます。

```
sp_text_dump_database ONLY_INDEXES
```

例 2 現在のデータベースが */data/db1backup* にダンプされてから、テキスト・インデックスがバックアップされます。

```
sp_text_dump_database CURRENT_DB_AND_INDEXES, "to '/data/db1backup'"
```

例 3 *text_db* データベースが */data/textdbbackup* にダンプされてから、テキスト・インデックスがバックアップされます。

```
sp_text_dump_database @backkupdbs = "TEXT_DB_AND_INDEXES",
@textdb_to = "to '/data/textdbbackup'"
```

例 4 現在のデータベースが */data/db1backup* に、*text_db* データベースが */data/textdbbackup* にそれぞれダンプされてから、テキスト・インデックスがバックアップされます。

```
sp_text_dump_database @backupdbs = "INDEXES_AND_DATABASES",
@current_to = "to '/data/db1backup'",
@textdb_to = "to '/data/textdbbackup'"
```

使用法

- 拡張型全文検索エンジンは *current_to*、*current_with*、*current_stripe01* ～ *current_stripe31* の値を dump database *currentdbname* に連結し、dump database コマンドを実行します。dump database コマンド実行時の出力は、拡張型全文検索エラー・ログに送信されます。
- 拡張型全文検索エンジンは、*textdb_to*、*textdb_with*、*textdb_stripe01* ～ *textdb_stripe31* の値を "dump database *currentdbname*" に連結し、dump database コマンドを実行します。dump database コマンド実行時の出力は、拡張型全文検索エラー・ログに送信されます。
- すべてのインデックスをバックアップするときに、現在のデータベースの text_events テーブル内にある「処理済み」ステータスを持つエントリはすべて削除されます。
- Verity コレクションのバックアップ・ファイルは、backDir 設定パラメータで指定したディレクトリに格納されます。
- バックアップのカスタマイズ方法については、設定パラメータ backCmd のリファレンスを参照してください。

メッセージ

- The parameter value '*value*' is invalid.
- Server name '*server*' does not exist in sysservers.
- Attempt to dump database '*database_name'* failed - use the dump database command.
- Attempt to backup text indexes on server '*server_name*' failed.
- Attempt to clean text_events in database '*database_name'* failed (date = '*date*').
- Parameter '*parameter_name*' is required when dumping database '*database_name'*.
- Dumping database '*database_name*' - check Full Text Search SDS error log for status.

パーミッション

すべてのユーザが、sp_text_dump_database を実行できます。

参照

『ASE リファレンス・マニュアル』の「dump_database」を参照してください。

sp_text_kill

説明

特定のテキスト・インデックスへの接続をすべて終了します。

構文

sp_text_kill *index_table_name*

パラメータ

index_table_name

すべての接続を終了するテキスト・インデックスの名前。*index_table_name* は [dbname.[owner.]]table の形式をとります。各パラメータの意味は、次のとおりです。

- dbname はインデックス・テーブルが入っているデータベースの名前です。owner またはプレースホルダがある場合、それも指定する必要があります。
- owner はインデックス・テーブルの所有者の名前です。
- table はインデックス・テーブルの名前です。

例

テキスト・インデックス i_blurbs への既存の接続をすべて終了します。

```
sp_text_kill "i_blurbs"
```

使用法

- このシステム・プロシージャによって拡張型全文検索エンジンは、指定されたインデックスへのすべての接続 (ただし、要求を開始した接続を除く) を終了します。
- 現在使用中のテキスト・インデックスを削除しようとすると失敗します。sp_text_kill を使用して既存の接続をすべて終了させると、インデックスを正常に削除できるようになります。

メッセージ

- Index ‘*index_table_name*’ is not a text index.
- This procedure is not supported against remote server ‘*server_name.*’
- ‘*index_table_name*’ does not exist.
- Only the System Administrator (SA) may execute this procedure.

パーミッション　ユーザ “sa” だけが、sp_text_kill を実行できます。

参照　sp_drop_text_index

sp_text_load_index

説明　テキスト・インデックスのバックアップをリストアします。

構文　sp_text_load_index

パラメータ　なし

例

```
sp_text_load_index
```

現在のデータベース内のテキスト・インデックスをすべてリストアします。

使用法

- sp_text_load_index を text_db データベースに対して実行し、現在のデータベースを完全にリカバリします。
- sp_text_load_index は最新のバックアップから Verity コレクションをリストアします。次に、拡張型全文検索エンジンは sp_redo_text_events と sp_text_notify を実行して、text_events テーブル内のインデックスをバックアップした日時以降のすべてのエントリを再適用します。

メッセージ

- Server name ‘*server_name*’ does not exist in sysservers.
- Unable to restore text indexes for server ‘*server_name.*’
- This procedure is not supported against remote server ‘*server_name*’
- Update to text_events table in database *database_name* failed for server ‘*server_name’* -text_events not rolled forward.

パーミッション　すべてのユーザが、sp_text_load_index を実行できます。

参照　sp_redo_text_events; sp_text_notify

sp_text_notify

説明

text_events テーブルが修正されたことを拡張型全文検索エンジンに通知します。

構文

sp_text_notify [{true | false}] [, *server_name*]

パラメータ

true
プロシージャを、同期をとりながら実行します。

false
プロシージャを非同期で実行します。

server_name
通知先の拡張型全文検索エンジンの名前。

例

```
sp_text_notify true
```

使用法

- 送信元テーブルが修正されたことを拡張型全文検索エンジンに通知するために、sp_refresh_text_index を発行したら sp_text_notify を実行してください。
- true も false も指定しない場合、sp_text_notify は同期をとりながら動作します。
- サーバ名を指定しない場合は、すべての拡張型全文検索エンジンに通知します。

メッセージ

- Can't run sp_text_notify from within a transaction.
- Notification failed, server = '*server_name.*'
- Server name '*server_name*' does not exist in sysservers.
- The parameter value '*value*' is invalid.

パーミッション

すべてのユーザが、sp_text_notify を実行できます。

参照

sp_refresh_text_index

sp_text_online

説明　データベースを Adaptive Server から全文検索できるようにします。

構文　sp_text_online [*server_name*], [*database_name*]

パラメータ

server_name
　拡張型全文検索エンジンの名前。

database_name
　オンラインにするデータベースの名前。

例　pubs2 データベースを拡張型全文検索エンジンで全文検索できるようにします。

```
sp_text_online @database_name = pubs2
```

使用法

- データベースが指定されない場合、すべてのデータベースを全文検索用にオンラインにします。
- サーバ名が指定されない場合、vesaux テーブルにリストされているすべての拡張型全文検索エンジンに通知します。
- 拡張型全文検索エンジンでは、auto_online 設定パラメータが 1 に設定されている場合、データベースは自動的にオンラインになります。

メッセージ

- All Databases using text indexes are now online
- Databases containing text indexes on server '*database_names*' are now online.
- Server name *server_name* is now online.
- Server name '*server_name*' does not exist in sysservers.
- The parameter value '*value*' is invalid.
- The specified database does not exist.
- vs_online failed for server '*server_name.*'

パーミッション　すべてのユーザが、sp_text_online を実行できます。

付 録 B

サンプル・ファイル

この付録の内容は次のとおりです。

- デフォルトの設定ファイル (*textsvr.cfg*) のテキスト
- sample_text_main.sql サンプル・スクリプトの概要
- 拡張型全文検索エンジンに用意されているすべてのサンプル・ファイルのリスト
- getsend プログラムの概要

デフォルトの *textsvr.cfg* 設定ファイル

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;  @(#)  File: textsvr.cfg  1.17  07/26/99
;
;  Full Text Search Specialty Data Store
;             Sample Configuration File
;
;  The installation procedure places this file in the
;  "SYBASE" directory.
;
;  Lines with a semi-colon in column 1 are comment lines.
;
;  Modification History:
;  ---------------------
;  11-21-97     Create file for Full Text Search SDS
;  03-02-98     Add trace flags and config values for
;               Enhanced Full Text Search SDS
;  05-26-99     remove references to sds/text
;  07-09-99     added batch block size
;  08-24-99    remove version string and correct copyright
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;             copyright (c) 1997, 1999
;             Sybase, Inc. Emeryville, CA
;             All rights reserved.
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;                               DIRECTIONS
```

```
;
;  Modifying the textsvr.cfg file:
;  -------------------------------
;  An installation can run the Text Search SDS product
;  as supplied, with no modifications to configuration
;  parameters. Default values from the executable program
;  are in effect.
;
;  The "textsvr.cfg" file is supplied with all configuration
;  parameters commented out.
;
;  The hierarchy for setting configuration values is:
;
;   default value internal to the executable program (lowest)
;   configuration file value  (overrides default value)
;   command line argument     (overrides default value and *.cfg file)
;
;  Command line arguments are available to override
;  settings for these options:
;
;     -i<file specification for interfaces file>
;     -l<file specification for log file>
;     -t (no arg) directs text server to write start-up
;        information to stderr (default is DO NOT write start-up information)
;
;  To set configuration file parameters, follow these steps:
;
;  (1) If changing the server name to other than "textsvr":
;      (1A)  Copy "textsvr.cfg" to "your_server_name.cfg"
;            Example: text_server.cfg
;      (1B)  Modify the [textsvr] line to [your_server_name]
;            Example: [text_server]
;      The maximum length of "your_server_name" is 30 characters.
;
;  (2) Set any configuration values in the CONFIG VALUES SECTION below.
;      Remove the semi-colon from column 1.
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;
;                   DEFINITIONS OF TRACE FLAG AND SORT ORDER VALUES
;
; "traceflags" parameter, for text server
;  Available "traceflags" values: 1,2,3,4,5,6,7,8,9,10,11,12,13
;
;  1  trace connect/disconnect/attention events
;  2  trace language events
;  3  trace rpc events
;  4  trace cursor events
;  5  log error messages returned to the client
;  6  trace information about indexes
```

```
;  7  trace senddone packets
;  8  write text server/Verity api interface records to the log
;  9  trace sql parser
; 10  trace Verity processing
; 11  disable Verity collection optimization
; 12  disable returning of sp_statistics information
; 13  trace backup operations (Enhanced Full Text Search only)
;
; "srv_traceflags" parameter, for Open Server component of text server
; Available "srv_traceflags" values: 1,2,3,4,5,6,7,8
;  1  trace TDS headers
;  2  trace TDS data
;  3  trace attention events
;  4  trace message queues
;  5  trace TDS tokens
;  6  trace open server events
;  7  trace deferred event queue
;  8  trace network requests
;
; "sort_order" parameter
; Available "sort_order" values: 0,1,2,3
;  0  order by score, descending (default)
;  1  order by score, ascending
;  2  order by timestamp, descending
;  3  order by timestamp, ascending
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;                       CONFIG VALUES SECTION
;
; The "textsvr.cfg" file is supplied with the values commented out.
; To override value(s) in the executable program:
;    - Set required value(s) below
;    - Remove the semicolon from column 1
;


[textsvr]
;min_sessions = 10
;max_sessions = 100
;batch_size = 500
;sort_order = 0
;defaultDb = text_db
;errorLog = textsvr.log
;language = english
;charset = iso_1
;vdkLanguage =
;vdkCharset = 850
;traceflags = 0
;srv_traceflags = 0
;max_indexes = 126
```

```
;max_packetsize = 2048
;max_stacksize = 34816
;max_threads = 50
;collDir = <txtsvr directory tree location on UNIX>/collections
;collDir = <txtsvr directory tree location on Win-NT>¥collections
;vdkHome = <txtsvr directory tree location on UNIX>/verity
;vdkHome = <txtsvr location on Win-NT>¥verity
;interfaces = <$SYBASE location on UNIX>/interfaces
;interfaces = <%SYBASE% location on Win-NT>¥ini¥sql.ini
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; The parameters in this section apply only to the Enhanced Full Text Search SDS.
; If defined to a Full Text Search engine they will be ignored.
;
;auto_online = 0
;backDir = <txtsvr directory tree location on UNIX>/backup
;backDir = <txtsvr directory tree location on Win-NT>¥backup
;backCmd =
;restoreCmd =
;knowledge_base =
;nocase = 0
;cluster_max = 0
;cluster_order = 0
;cluster_style = Fixed
;cluster_effort = Default
;batch_blocksize = 0
;max_session_fd = 0
```

sample_text_main.sql スクリプト

拡張型全文検索エンジンのインストールでは、sample_text_main.sql スクリプトが、*$SYBASE/$SYBASE_FTS/sample/scripts* ディレクトリにコピーされます。このスクリプトは、以下のオペレーションを使用する方法を示しています。

- テキスト・インデックスの設定。
- データの修正と、コレクションへの変更内容の伝達。このオペレーションには、挿入、更新、削除が含まれる。
- テキスト・インデックスの削除。

インストールまたは設定のためにこのスクリプトを実行する必要はありません。このスクリプトは、サンプルとして提供されているものです。

sample_text_main.sql スクリプトを実行する前に必要になることは次のとおりです。

- Adaptive Server と拡張型全文検索エンジンが設定され、稼働していること。
- テキスト・エディタを使用して、sample_text_main.sql スクリプトを編集すること。"YOUR_TEXT_SERVER" を、sample_text_main.sql スクリプトの手順 4 の拡張型全文検索エンジンの名前に変更します。
- model データベースに text_events テーブルがあること。model データベースがそのように設定されていなければ、次の処理が必要です。
  - データベースを作成したら終了するように、sample_text_main.sql スクリプトを修正すること。
  - installevent スクリプトを新しいデータベースに適用すること。詳細については、「installevent スクリプトの実行」(15 ページ) を参照してください。
  - サンプル・スクリプトの残りを実行すること。

スクリプトを Adaptive Server への入力として指定します。たとえば、MYSVR という名前の Adaptive Server で sample_text_main.sql スクリプトを実行するには、次のように入力します。

```
isql -Ulogin -Ppassword -SMYSVR
-i
$SYBASE/$SYBASE_FTS/sample/scripts/sample_text_main.sql -
omain.out
```

このサンプル環境が必要なくなったら、Adaptive Server にログインして、サンプル・データベースを削除します。次に例を示します。

```
1> use master
2> go
1> drop database sample_colors_db
2> go
```

sample_text_main.sql は必要に応じて再実行できます。

拡張型全文検索エンジンの機能を示すサンプル・ファイル

拡張型全文検索エンジンには、Text Server のオペレーションを示す一連のサンプル・ファイルが用意されています。これらのファイルは、*$SYBASE/$SYBASE_FTS/sample/scripts* ディレクトリにあります。サンプル・ファイルの実行は、拡張型全文検索エンジンのインストール、設定、またはオペレーションには必要ありません。

カスタム・シソーラス

次のファイルは、カスタム・シソーラスの設定および使用方法について説明しています。

- *sample_text_thesaurus.ctl* － 制御ファイルのサンプル
- *sample_text_thesaurus.sql* － 制御ファイルのサンプルによって作成したカスタム・シソーラスを使用するクエリのサンプル

トピック

次のファイルは、トピックの設定および使用方法について説明しています。

- *sample_text_topics.otl* － アウトライン・ファイルのサンプル
- *sample_text_topics.kbm* － 知識ベース・マップのサンプル
- *sample_text_topics.sql* － 定義されたトピックを使用するクエリのサンプル

クラスタリング、要約、QBE (例示による問い合わせ)

次のファイルは、クラスタリング、要約、QBE の設定および使用方法について説明しています。

- *sample_text_setup.sql* － サンプル環境の作成
- *sample_text_queries.sql* － その環境に対するクエリの発行と環境の削除

getsend サンプル・プログラム

拡張型全文検索エンジンには、text データや image データをファイルから Adaptive Server で定義されているカラムにロードするための getsend というプログラムが含まれています。

必要なソース・ファイルとヘッダ・ファイル、makefile、プログラムを作成し実行するための指示は、*$SYBASE/$SYBASE_FTS/sample/source* ディレクトリにあります。

プログラムの使用方法については、*README.TXT* ファイルと *getsend.c* ファイルを参照してください。

付 録　C

Unicode のサポート

Unicode 標準は、ISO (国際標準化機構) による ISO 10646 標準のサブセットでもある国際文字セットです。Unicode は ISO 10646 の BMP (Basic Multilingual Plane) と一致するもので、世界の主要な文字と言語をすべてサポートします。したがって、Unicode は既存のすべての文字セットのスーパーセットです。

Unicode の主な利点は次のとおりです。

- 単一ソースによる開発が可能です。つまり、一度アプリケーションを開発すると、それを複数のロケール用に複数の言語へとローカライズできます。一体化された単独の文字セットを使用することで、複数の文字セット間の差異を考慮するためのアプリケーション修正が必要なくなり、開発、テスト、およびサポートのコストを削減できます。
- 1 つのデータベースで複数の言語が使用可能です。すべて Unicode 化されたシステムでは、データの文字セット判別を意識したデータベースを設計する必要はありません。

拡張型全文検索エンジンは Unicode をサポートしています。この機能を使用するには、charset 設定値を utf8 に設定します。これには、Unicode 対応のクライアント / サーバ・データベース・システムの構築に必要なものがすべて含まれています。

データを Unicode フォーマットで格納するように拡張型全文検索エンジンを設定するには、設定値 charset を utf8 に設定します。詳細については、「設定パラメータの修正」(62 ページ) を参照してください。

注意 Unicode フォーマットのデータに対してワイルドカード検索を行う場合は、トレース・フラグ 15 をオンにします。詳細については、「トレース・フラグの設定」(68 ページ) を参照してください。

付 録　D

XML データの使用

この付録の内容は次のとおりです。

- フィールドとゾーンで XML データを使用する場合の正しいフォーマット
- フィールドとゾーンを使用してテキスト・インデックスにインデックス付けした XML データのサンプル

フィールドとゾーンでの XML データの正しいフォーマット

Adaptive Server データベースのテキスト・カラムに XML データを挿入し、そのデータにテキスト・インデックスを作成できます。そのためには、Verity 演算子を含む特別な構文を伴う select 文を使用する必要があります。

たとえば、次の select 文では、<in> が Verity 演算子です。

```
select t1.id
from
    ti_address_tbl t1,
    address_tbl t2
where
    t1.id=t2.id and
    t1.index_any ='USA <in> address'
```

Sybase では、XML ドキュメントの形式が正しく、次のフォーマットを使用している場合に、フィールドまたはゾーンを使用してテキスト・インデックスに XML データをインデックス付けできます。

```
<address>
<street>123 Main St.</street>
<city>Anywhere</city>
<state>CA</state>
<country>USA</country>
</address>
```

このフォーマットでは、“123 Main St. Anywhere CA USA” というテキストを address ゾーンに挿入し、“123 Main St.” というテキストを street ゾーンに挿入しています。それ以降も同様です。

XML のインデックス付けのサンプル

この項では、フィールドとゾーンでの XML データの正しいフォーマットに記載のフォーマットを使用する例を示します。このサンプルでは、次の手順を実行します。

1 空のテーブルに空のテキスト・インデックスを作成します。

2 テキスト・インデックスを削除します。この手順では、Adaptive Server のテキスト・インデックスに関連付けられているプロキシ・テーブルは削除しますが、EFTS テキスト・インデックスの collections ディレクトリ構造内のデフォルトのディレクトリとファイルはすべて残します。

3 style.dft ファイルを変更してユニバーサル・フィルタを設定します。

4 テーブルにデータを挿入します。

5 テキスト・インデックスを再作成します。

6 テキスト・インデックスを使用して、XML データに基づいてローを分離する select 要求を発行します。

セクション別のサンプル

セクション 1

このセクションでは、空のテーブルに空のテキスト・インデックスを作成しています。

```
isql -Sase1501sunbox -Usa -P -w2048 Dxml_db1

create table address_tbl

        (
                id numeric(5,0)
                identity, xmlcol text
        )
go
create unique index uidx on address_tbl(id)
go
sp_create_text_index 'textsvr', 'ti_address_tbl',
'address_tbl', "empty ", 'xmlcol'
go
(return status = 0)
quit
```

セクション 2

このセクションでは、空のテキスト・インデックスを削除しています。この例では、Adaptive Server のテキスト・インデックスに関連付けられているプロキシ・テーブルは削除しますが、EFTS テキスト・インデックスの collections ディレクトリ構造内のデフォルトのディレクトリとファイルはすべて残します。

```
isql -Sase1501sunbox -Usa -P -w2048 Dxml_db1
sp_drop_text_index 'address_tbl.ti_address_tbl'
go
(return status = 0)
quit
```

セクション 3

このセクションでは、style.dft ファイルを変更してユニバーサル・フィルタを設定しています。

```
/sy/ase1501sunbox/EFTS-15_0/collections
          xml_db.dbo.ti_address_tbl/style
(uid=syuid)_sunbox> pwd
/sy/ase1501sunbox/EFTS-15_0/collections
          xml_db.dbo.ti_address_tbl/style
(uid=syuid)_sunbox>ls -al
Total 74
drwxr-xr-x 2 sybase sybase
512 Jan  8 16:14 .
drwxr-xr-x  11 sybase sybase
512 Jan  8 16:14 ..
...
-rw-r--r-- 1 sybase sybase
288 Jan  8 16:14 style.dft
...
(uid=syuid)_sunbox> mv ./style.dft
./style.dft.orig
(uid=syuid)_sunbox> cp
./style.dft.orig ./style.dft
(uid=syuid)_sunbox> vi ./style.dft
(uid=syuid)_sunbox> diff
./style.dft.orig ./style.dft
11a12
> /filter="universal"
(uid=syuid)_sunbox>
```

セクション 4

このセクションでは、テーブルに XML データを挿入しています。

```
isql -Sase1501sunbox -Usa -P -w2048 -Dxml_db
insert address_tbl values
('<?xml version="1.0"
encoding="UTF-8"?>
<fulldoc>
<address>
<street>mailstop
101</street>
<city>CONCORD</city>
<state>MA</state>
<country>USA</country>
</address>
<filepath>
/sy/ase1501sunbox/EFTS-15_0/collections
/made_using_mkvdk
/xml_data
/row1_well_formed.xml

</filepath>
</fulldoc>
')
go
(1 row affected)
insert address_tbl values
('<?xml version="1.0"
encoding="UTF-8"?>
<fulldoc>
<address>
<street>building 6</street>
<city>BALTIMORE</city>
<state>MD</state>
<country>USA</country>
</address>
<filepath>
/sy/ase1501sunbox/EFTS-
15_0/collections
/made_using_mkvdk/xml_data/row2_well_formed.xml
</filepath>
</fulldoc>
')
go
(1 row affected)
quit
```

セクション 5

このセクションでは、テキスト・インデックスを再作成しています。

```
isql -Sase1501sunbox -Usa -P -w2048 Dxml_db1
sp_create_text_index 'textsvr'
'ti_address_tbl', 'address_tbl', " ", 'xmlcol'
go
(return status = 0)
quit
```

セクション 6

このセクションでは、テキスト・インデックスを使用して、XML データに基づいてローを分離する select 要求を発行しています。

```
isql -Sase1501sunbox -Usa -P -w2048 -Dxml_db
select
    t1.id
from
    ti_address_tbl t1,
    address_tbl 2
where
    t1.id = t2.id and
    t1.index_any = 'USA <in> address'
go
quickpass disabled, reason: local table
SELECT
    id
FROM
    xml_db.dbo.ti_address_tbl
WHERE
    (
        index_any = 'USA <in> address'
    )
id
--------------
1
2
(2 rows affected)
quit


isql -Sase1501sunbox -Usa -P -w2048 -Dxml_db
select
    t1.id
from
    ti_address_tbl t1,
    address_tbl t2
where
    t1.id=t2.id and
    t1.index_any ='USA <in> country'
```

```
go
quickpass disabled, reason: local
table
SELECT
    id
FROM
    xml_db.dbo.ti_address_tbl
WHERE
    (
        t1.index_any = 'USA <in> country'
    )
id
-----------------
1
2
(2 rows affected)
quit


isql -Sase1501sunbox -Usa -P -w2048 -Dxml_db
select
    t1.id
from
    ti_address_tbl t1,
    address_tbl t2
where
    t1.id=t2.id and
    t1.index_any='USA'
go
quickpass disabled, reason: local table
SELECT
    id
FROM
    xml_db.dbo_ti_address_tbl
WHERE
    (
        t1.index_any ='USA'
    )
id
---------------
1
2
(2 rows affected))
quit


isql -Sase1501sunbox -Usa -P -w2048 -Dxml_db
select
    t1.id
from
    ti_address_tbl t1,
    address_tbl t2
where
```

```
    (
    t1.id=t2.id and
    t1.index_any='mailstop <in> address <and>
    MA <in> state'
go
quickpass disabled, reason: local table
SELECT
    id
FROM
    xml_db.dbo_ti_address_tbl
WHERE
    (
        index_any ='mailstop <in> address <and> MA
    <in> state'
    )
id
---------------
1
(1 row affected))
quit
```

```
isql -Sase1501sunbox -Usa -P -w2048 -Dxml_db
select
    t1.id
from
    to_address_tbl t1
    address_tbl t2
where
    t1.id=t2.id and
    t1.index_any =
    'sy <in> filepath <and>
    asesunbox <in> filepath <and>
    EFTS-15_0 <in> filepath
    <and> collections <in> filepath <and>
    made_using_mkvdk <in> filepath <and>
    xml_data <in> filepath <and> ro* <in> filepath'
go
quickpass disabled, reason: local table
SELECT
    id
FROM
    xml_db.dbo.ti_address_tbl
WHERE
    (
        index_any = 'sy <in> filepath <and>
        ase1501sunbox <in> filepath <and>
        EFTS-15_0 <in> filepath
        <and> collections <in> filepath <and>
        made_using_mkvdk <in> filepath <and>
```

```
        xml_data <in> filepath <and> ro* <in> filepath'
id  )
----------
1
2
(2 rows affected)
quit



isql -Sase1501sunbox -Usa -P -w2048 -Dxml_db
select
    t1.id
from
    ti_address_tbl t1,
    address_tbl 2
where
    t1.id = t2.id and
    t1.index_any =
        'sy <in> filepath <and>
        ase1501sunbox <in> filepath <and>
        EFTS-15_0 <in> filepath
        <and> collections <in> filepath <and>
        made_using_mkvdk <in> filepath <and>
        xml_data <in> filepath <and> row2* <in> filepath'
go
quickpass disabled, reason: local table
SELECT
    id
FROM
    xml_db.dbo.ti_address_tbl
WHERE
    (
        index_any = 'sy <in> filepath <and>
        ase1501sunbox <in> filepath <and>
        EFTS-15_0 <in> filepath <and>
        collections <in> filepath <and>
        made_using_mkvdk <in>
        filepath <and>
        row2* <in> filepath')
id
----------
2
(1 row affected)
quit
```

索引

記号

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

あ

い

え

お

か

き

く

け

こ

さ

し

す

せ

そ

た

ち

て

と

ね

は

ひ

ふ

め

も

や

ゆ

よ

り

れ